VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYNCHRONIZACE KONTAKTŮ V PRIVÁTNÍM CLOUDU

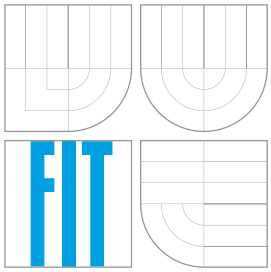
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAROSLAV SENDLER

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYNCHRONIZACE KONTAKTŮ V PRIVÁTNÍM CLOUDU

CONTACT SYNCHRONIZATION FOR A PRIVATE CLOUD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAROSLAV SENDLER

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. ROMAN TRCHALÍK, Ph.D.

BRNO 2014

Abstrakt

Tato diplomová práce se zabývá možností vytvoření vlastního synchronizačního nástroje pro zařízení s operačním systémem Android. Jde o možnost synchronizace kontaktů z přenosných zařízení v privátním cloudu, který je zde substituován adresářovým serverem. Pro práci s adresářovým serverem je použit protokol LDAP a jeho serverová implementace OpenLDAP. První část práce je zaměřena na popis a strukturu adresářových služeb, ať už jde o způsob uložení dat nebo o nabízené funkce. Následuje analýza a vyčlenění požadavků na samotnou aplikaci. Celou práci ukončují návrhy a implementace týkající se bezpečnosti a pokročilé možnosti synchronizace pouze vybraných kontaktů.

Abstract

This masters's thesis is studying the possibility of Create a synchronization tool for device running on Android. It is the ability to sync contacts from portable devices in a private cloud, which is here substituted by Directory server. For working with directory, server uses LDAP server and its implementation of OpenLDAP. The first part is aimed at the description of the structure and directory services. Following is analysis and separation requirements. The whole project is completed by proposals and implementation that relate to the safety and advanced synchronization.

Klíčová slova

Synchronizace, cloud, adresářové služby, OpenLDAP, Android, SSL/TLS, UnboundID.

Keywords

Synchronization, cloud, Directory service, OpenLDAP, Android, SSL/TLS, UnboundID.

Citace

Jaroslav Sendler: Synchronizace kontaktů v privátním cloudu, diplomová práce, Brno, FIT VUT v Brně, 2014

Synchronizace kontaktů v privátním cloudu

Prohlášení

Prohlašuji, že jsem tuto diplomouvou práci vypracoval samostatně pod vedením pana Mgr. Romana Trchalíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Sendler
27. května 2014

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce Mgr. Romanovi Trchalíkovi, Ph.D, který mi poskytl odbornou pomoc při jejím vypracování a za ochotu a kladný přístup při konzultacích.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době se setkáváme s mnoha automatizovanými procesy, které se nám snaží ulehčit a zpříjemnit práci. Za určitých podmínek a úhlů pohledu lze k této kategorii přiřadit i funkci zvanou synchronizace. S pojmem cloud dnes tvoří silnou dvojici, která dokáže zaujmout mnoho uživatelů.

Koncepce cloudu měla v nedávné době velkou mediální podporu, přestože je zde s námi v určité formě již několik let. Až v posledních letech, kdy mu bylo přiřazeno nové zvučné marketingové jméno, a se zkvalitněním internetových připojení a rozšířením jejich dostupnosti, se tento pojem dostává i do povědomí veřejnosti. Tyto podmínky poskytly ideální prostředí pro zaplnění díry na trhu a tak začaly vznikat nové služby, které se snaží uživatelům poskytnout inovované funkce. Jejich hlavní náplní by mělo být ulehčení a zpříjemnění práce nejen na počítačích. Ať už se jedná o různá datová úložiště, kalendáře nebo o další produkty, vždy v těchto případech uživatel svěřuje svá osobní data některé cizí straně poskytující danou službu. To představuje riziko zneužití osobních dat třetí stranou a nemožnost mít plnou kontrolu nad svými daty.

S nástupem chytrých telefonů (smartphone) se pozice cloudu a synchronizace ještě zvýšila. Vezme-li se v potaz, že mobilní zařízení se stalo neoddělitelnou součástí našich životů, tak nikoho nepřekvapí, že jejich paměti uchovávají citlivá data. Z obav odcizení či ztráty informací se mnoho uživatelů obrací na odbornou pomoc. Ta nabízí možnost zálohy uživatelových dat všeho druhu.

Cílem této diplomové práce je především poskytnutí nástroje k nahrazení stávající možnosti zálohování kontaktů v mobilních zařízení. Měla by sloužit uživatelům, kteří nechtějí svá data poskytovat třetí straně a o bezpečnost a správu svých informací se chtějí starat sami. Předmětem této práce je vytvoření řešení zálohování kontaktů na základě vlastních prostředků a to bez použití externích služeb jako je například synchronizace od firmy Google. Řešení se skládá ze samotné mobilní aplikace a adresářové služby.

Privátně je aplikace navržena pro synchronizaci kontaktů, tedy údajů o různých uživatelích jako je jméno, příjmení, telefon, email a další. Aplikace je připravena pro operační systém Android, zejména kvůli jeho rozšířenosti mezi uživateli. Synchronizace bude zajištěna pomocí protokolu LDAP. Pro serverovou část bude použita open sourcová implementace LDAP protokolu, software OpenLDAP. S využitím této aplikace, tedy nainstalováním ji na přenosné zařízení a nastavení LDAP serveru, dostane uživatel plnohodnotnou službu pro synchronizaci kontaktů v privátním cloudu ve své vlastní režii.

K základním funkčnostem aplikace se řadí možnost připojit se k nakonfigurovanému serveru podporující LDAPv3 a synchronizovat s tímto adresářovým serverem předem specifikovaná data. Kontakty bude možné pomocí mobilní aplikace nejen vyhledávat, ale i editovat,

mazat a v určitém módu i vytvářet nové. K pokročilejším funkcím patří možnost synchronizace pouze určitých uživatelů. Kontakty, které budou synchronizovány, bude možné určit ručním výběrem ze seznamu nebo splněním různých podmíněných kritérií jako je účast v dané skupině a další. Tímto způsobem bude možné oddělit privátní data od korporátních a každý uživatel si bude moci určit, které kontakty budou synchronizovány.

Diplomový projekt je tvořen z šesti kapitol. Kapitola druhá popisuje a shrnuje základní informace o protokolu LDAPv3, na kterém je postavena serverová část projektu. Jde především o popis následujících modelů: informační, jmenný, funkční a bezpečnostní. Třetí kapitola je zaměřena na získání a upřesnění požadavků na celý vyvíjený systém, kde se první část zaměřuje na analýzu a vyčleňuje základní poznatky. Také jsou zde rozebrána již stávající řešení a definovány funkční i nefunkční požadavky. V kapitole čtvrté, nazvané návrh, je možné nalézt obecný návrh aplikace s jeho popisem a návrh řešení bezpečnosti. Zaměřuje se na popis využití databází jak pro získání dat, tak pro uložení dodatečných informací. Jsou zde uvedeny diagramy a popis případu užití. K nejzajímavějším částem této kapitoly lze přiřadit část, která se zabírá výběrem vhodného algoritmu a jeho samotným rozborem. Závěr kapitoly je zaměřen na vývoj grafického uživatelského rozhraní. Konkrétně jde o návrh obrazovek a jejich návaznosti. Čtvrtá část diplomové práce je věnována implementaci. Jsou zde rozebrány a popsány základní stavební kameny aplikace i podpůrné nástroje využité při psaní kódu. Kapitola je uzavřena tématem tvorby LDAP schématu a adresářové struktury. Poslední kapitola, závěr, shrnuje dosažené výsledky a navrhuje další pokračování.

Kapitola 2

LDAP

V následující kapitole jsou, pro ujasnění a pochopení dalších částí práce, rozebrány základní stavební kameny protokolu Lightweight Directory Access Protocol (LDAP). Adresářová služba LDAP je vybrána pro ukládání a následně také pro získávání informací, tedy supluje zde cloudové úložiště. Konkrétně jsou zde popsány stávající vlastnosti implementace, architektura protokolu LDAP obecně [9, 8, 12] a jiné detaily [7]. Další informace použité pro popis jsou čerpány z jednotlivých RFC[1] dokumentů, které definují základní kostru LDAP [17].

LDAP je TCP/IP protokol pracující na bázi klient/server, původně založený na podmnožině X.500 Directory Access Protocol (DAP). Vznikl ze snahy zjednodušit X.500, který je složen z celé řady standardů (ITU Telecommunication Standardization Sector) popisujících všechny aspekty globální adresářové služby. X.500 nebyl nikdy ve větší míře realizován, příčinu tohoto dopadu je možné přiřadit jeho implementační náročnosti.

LDAP není jediná adresářová služba, do této kategorie jsou začleněny i další. Jako příklad dnes běžně používaných lze uvést službu od společnosti Novell eDirectory nebo od společnosti Microsoft Active Directory a další. Všechny služby jsou postaveny na zajištění přístupu k datům typu adresář, který lze chápat jako jednoduchou kolekci informací, avšak lišící se od významu v souborovém systému. Lze jej přirovnat k tzv. žlutým stránkám[2] nebo bílým stránkám[3].

LDAP využit v této práci je ve verzi 3. Nižší verzi (LDAPv2) již není doporučeno používat [24] a to zejména z důvodu neposkytování odpovídajících bezpečnostních prvků, kódování textových hodnot a dalších omezení.

[1] RFC request of comments — Žádost o komentáře
[2] Yellow Pages — telefonní seznamy firem
[3] White Pages — telefonní seznam domácností

V níže uvedené tabulce 2.1 jsou vypsány základní standardy, které jsou aktuálně doporučeny pro LDAPv3. První sloupec obsahuje číslo RFC dokumentu a druhá část jeho název. Z těchto dokumentů vychází i samotná diplomová práce.

RFC	Název
RFC 4511 [17]	The Protocol
RFC 4512 [21]	Directory Information Models
RFC 4513 [10]	Authentication Methods and Security Mechanisms
RFC 4514 [23]	String Representation of Distinguished Names
RFC 4515 [19]	String Representation of Search Filters
RFC 4516 [20]	Uniform Resource Locator
RFC 4517 [13]	Syntaxes and Matching Rules
RFC 4518 [22]	Internationalized String Preparation Top
RFC 4519 [16]	Schema for User Applications

Tabulka 2.1: LDAP standardy.

V dalších částech kapitoly budou rozebrány jednotlivé modely, z nichž je LDAP tvořen. Jde konkrétně o model informační, jmenný, funkční a bezpečnostní.

Některé příklady a obrázky používané v této kapitole jsou inspirovány nebo přímo využívají LDAP schéma, které je použito na fakultě informačních technologií v Brně[4].

2.1 Informační model

Informační model LDAP poskytuje struktury a typy dat, které mohou být ukládány do adresáře. Lze říci, že informační model definuje základní stavební kameny, na kterých je postavena tvorba adresářů a samotná struktura informačního adresářového stromu (více o DIT v kapitole 2.2.1). K primárním blokům patří záznamy, atributy a hodnoty, které dohromady tvoří adresářové schéma.

Základní jednotkou, která sdružuje kolekci informací o objektech, je záznam. LDAP adresáře převážně uchovávají informace z běžného světa kolem nás a právě tyto hodnoty jsou umístěny v záznamech. Záznam je tvořen množinou atributů, kde každý popisuje jednu konkrétní vlastnost objektu. Každý z atributů je určen jeho typem a jednou jeho vlastnosti nebo vlastnostmi, které bude mít.

Příklad jednoduchého záznamu je ukázán v tabulce 2.2, která se skládá ze dvou sloupců. První sloupeček znázorňuje jednotlivé atributy a druhý obsahuje jejich hodnoty. Tabulka popisuje záznam studenta vysoké školy s nejběžnějšími atributy jako jsou: jméno, příjmení, stát, email, adresa a další. V závorkách prvního sloupce se nachází zkratky nebo zkrácené názvy typů atributů, se kterými je možné se běžně setkat. Druhý sloupec obsahuje hodnoty atributů, což jsou data, která ukládáme do adresářů.

Tvorbu struktury LDAP lze přirovnat k většině novým programovacím jazykům. Tak jako převážná část moderních jazyků využívá objektové programování, tak i v LDAP adresářích je využito těchto technik. V následující části jsou představeny základní prvky objek-

[4]Server je dostupný na adrese *ldap.fit.vutbr.cz*.

tového prostředí a jejich použití. Jako příklad lze uvést dědičnost z jiných objektů, která je hojně využívaná při tvorbě nových schémat.

Typ atributy	Hodnota atributu
commonName (cn)	Jaroslav Sendler
name	Jaroslav
surname (sn)	Sendler
countryName (c)	Česká republika
streetAddress (street)	Božetěchova
localityName (l)	Brno
telephoneNumber	+420 XXX XXX XXX
mail	xsendl00@stud.fit.vutbr.cz

Tabulka 2.2: Záznam studenta VUT.

2.1.1 Třídy objektů

Třída objektů je rodina objektů, která sdílí určité charakteristiky, viz RFC 4512 [21]. Sdružuje a popisuje objekty se záznamy, které k nim náleží. Pomocí třídy objektů lze záznamům nadefinovat dvě různé skupiny atributů: povinné a volitelné. Není-li množina povinných atributů prázdná, je nutné, aby tyto atributy byly nenulové. Naopak volitelné atributy nejsou tímto způsobem nijak omezeny, jejich výskyt je dobrovolný. Ke kolekcím záznamům se přiřazují určité zásady, které řídí a kontrolují způsob, jakým se k záznamům přistupuje. Záznamy jsou uloženy v adresářích a díky třídám objektů lze kontrolovat a přiřazovat operace, které lze nad nimi provádět. S tím úzce souvisí i samotné umístění záznamů ve stromové struktuře. Protože ne každý objekt se může vyskytovat na kterékoliv pozici.

Třídy, tak jako moderní programovací jazyky, využívají základní objektovou vlastnost a to dědičnost. U LDAP při vytváření objektů adresáře lze využít nejen jednoduchou dědičnost jako v jazyce Java, ale i vícenásobnou dědičnost, jaká se využívá například v jazyce C++. Při vytváření objektů pomocí dědění je dodržováno obvyklé pravidlo, a to že se atributy nadřazené třídy dědí do třídy podřazené. V tomto případě, kdy existují dvě skupiny atributů: povinné a volitelné, jsou samozřejmě povinné atributy zděděny opět jako povinné. U druhé množiny se nabízí možnost přesunu volitelných atributů na povinné.

Při tvorbě objektu jsou používány tzv. třídy objektů. Existují tři základní druhy: abstraktní, strukturální a pomocné. Z abstraktní třídy se odvozují všechny strukturální třídy, na druhou stranu pomocné třídy z ní vycházet nemusí. Abstraktní třída se nazývá „top“ a poskytuje základní charakteristiky z nichž další třídy dědí. Slouží jako vzor pro tvorbu jiných tříd. Strukturální třída je určena k definování pozice záznamu v adresářovém informačním stromě a pro formulaci obsahu záznamu. Posledním typem tříd jsou tzv. pomocné třídy. Jsou používány k rozšíření charakteristik záznamů pomocí sady atributů a pro popis položek tříd.

Příklad běžně používané třídy je *person*. *Person* je třída z nichž vychází většina tříd, které mají představovat osoby a uchovávat informace o nich. Přesná definice třídy *person* je ukázána ve schématu 2.1, které je převzato z RFC 4519 [16].

Každý objekt třídy začíná řetězcem čísel, která jsou oddělena tečkou. Toto číslo tzv. OID

(object identifical) neboli identifikační číslo objektu, nesmí být svou strukturou v konfliktu s jinými objekty. V tomto případě je OID tvaru *2.5.6.6*. Po OID následuje název třídy (*person*), případně i jméno nadřazené třídy (*top*) a její typ (*STRUCTURAL*). Jméno objektu třídy musí být jednoznačné na serveru, kde je služba provozována. Zbylou část definice třídy *person* tvoří atributy. Povinné atributy v sekci *must* jsou celé jméno (commonName) a příjmení (surname). Oba tyto údaje musí obsahovat každý objekt typu person, naproti tomu atributy: heslo (userPassword), telefonní číslo (telephoneNumber), popis (description) a další (seeAlso), v sekci *may* jsou volitelné a mohou být vynechány.

```
( 2.5.6.6 NAME 'person'
          SUP top
          STRUCTURAL
          MUST ( sn $ cn )
          MAY ( userPassword $  telephoneNumber $ seeAlso $ description )
)
```

Obrázek 2.1: Definice třídy *person* z RFC 4519 [16].

Přesnou definici struktury třídy lze nalézt v RFC 4512 [21].

2.1.2 Atributy

Typy atributů jsou základní stavební bloky LDAP záznamů. Jsou součástí tříd, které plní roli schránky, která uchovává informace o objektech. Atributy jsou definovány následujícími komponentami: jménem, OID, syntaxí, množinou pravidel a pomocí metody děděním.

Jméno je řetězec znaků, který nerozlišuje velká a malá písmena. Slouží uživatelům ke snadnější orientaci mezi atributy. Hlavním rozlišovacím údajem je, tak jako u objektů tříd, OID, které ani zde nesmí být v konfliktu s jiným OID. Proto je možné použít dva různé názvy a namapovat je na jedno OID, jelikož se při práci se záznamy vždy využívá OID místo jména.

Z důvodu, že LDAP je síťový protokol a pro přenos informací po síti využívá 8-bitové řetězce, obsahuje každý atribut informaci o tom, jakého typu data udržuje. Tyto informace jsou definovány pomocí syntaxe, která určuje podobu dat, které mohou být atributem prezentovány. Jako příklad syntaxe lze uvést Directory String, Binary, Integers a další. I v těchto případech je využito číselné označení s tečkovou notaci, tedy OID. Pomocí syntaxe lze nadefinovat i určitá omezení, jako je velikost nebo délka či rozsah hodnot. Například lze tímto způsobem omezit velikost dat, které je možné atributem udržovat.

Přestože syntaxe definuje, jakým způsobem jsou data uložena v atributech, neříká nic o tom, jak s nimi pracují operace popsané ve funkčním modelu 2.3. I přesto má typ atributu vliv na to, jakým způsobem operace funguje. Je to díky porovnávacím pravidlům, které úzce souvisí s typem. Pravidla lze rozdělit do čtyř následujících skupin: shoda, řadící, podřetězec a podschéma. Již podle názvu je patrné k čemu jsou tato pravidla vhodná. Při snaze porovnání hodnoty od klienta s hodnotou atributu uloženou na serveru se použije pravidlo na shodu. Jako příklad lze uvést pravidlo pro atribut typu *Directory String*, *caseIgnoreMatch*. Jeho použití je vhodné pro textové řetězce, jelikož při porovnání nebere v úvahu rozdíl mezi malými a velkými písmeny. Řadící pravidla jsou používána k rozhodnutí, zda je jedna hodnota větší nebo menší než hodnoty jiné. Předposlední skupina se využívá v případech, ve kterých není nutná celá shoda hodnot, ale postačuje pouze jejich část neboli podřetězec.

Seznam některých pravidel, které patří k nejpoužívanějším, lze nalézt v tabulce 2.3. Tabulka je tvořena dvěma sloupci, kde u prvního je možné zjistit originální název pravidel a druhý obsahuje objekt porovnání. Obsah tabulky vychází z dokumentu RFC 4517 [13], kde je možné najít další pravidla a jejich podrobnější popis s příklady použití.

Název pravidla	**Typ hodnoty**
booleanMatch	Boolean
caseIgnoreMatch	Directory String
caseIgnoreOrderingMatch	Directory String
integerMatch	Integer
telephoneNumberSubstringsMatch	Telephone Number

Tabulka 2.3: Porovnávací pravidla nad atributy LDAP.

Atributy lze rozdělit na to zda mohou být vícehodnotové nebo naopak mohou mít pouze jednu hodnotu. Označí-li se atribut jako „jedno hodnotový“, znamená to, že se v objektu může vyskytovat pouze jedenkrát. Na druhou stranu pokud atributy reprezentují položky reálného světa, jednoduché atributy by na jeho popis nestačily. Jako příklad lze uvést atribut obsahující elektronickou poštu. Pro uložení těchto hodnot se jako vhodné jeví použití vícehodnotového typu atributů. Samozřejmě je lepší pro každou emailovou adresu vytvořit odpovídající jednoduchý atribut, ale ne vždy je toto možné. V případě, kde předem není znám přesný počet adres se bude tato možnost jevit jako nevhodná. Z tohoto důvodu se ve schématu vytvořeném pro tuto práci nachází mnoho atributů stejného typu, například *homeEmail*, *workEmail* a *otherEmail*.

Některé atributy nejsou navrženy pro to, aby byly modifikovány adresářovými aplikacemi například z mobilního telefonu. Tyto atributy, *provozní*, tvoří tzv. metadata. Jsou to interní informace o záznamech, které jsou modifikovány na základě práce s nimi a jsou převážně určeny pro administrátory. Pro tuto práci jsou nezbytné následující atributy:

1. **modifiersName** - neboli jednoznačné jméno (DN) uživatele, který tyto položky jako poslední upravoval. Položka se objeví u záznamů, které byly modifikované pomocí protokolu LDAP (aktualizační operací 2.3.3).

2. **modifyTimestamp** - neboli čas, kdy byl záznam naposledy upraven. Položka se objeví u záznamů, které byly modifikované pomocí protokolu LDAP. Podrobnější využití tohoto atributu bude rozebráno v části zabývající se synchronizací.

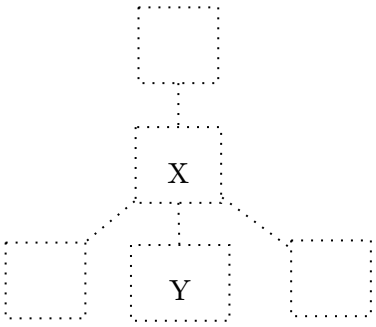
K některým dalším atributům patří: *creatorsName* a *createTimestamp*. Zbylé, zde neuvedené, je možné nalézt v RFC 4512 [21].

2.2 Jmenný model

Jmenný model definuje formát, jakým jsou jednotlivé položky organizovány a jak je možné se na ně jednoznačně odkazovat. Jako struktura pro ukládání dat v LDAP je zvolen strom, tzv. *Directory Information Tree*, který je popsán v následující části níže 2.2.1. Se strukturou stromu úzce souvisí pojmenování záznamů, které je rozebráno v dalších částech o *DN* a *RDN* 2.2.2.

2.2.1 DIT

DIT neboli Directory Information Tree je stromová datová struktura, která obsahuje položky hierarchicky organizované. Jednotlivé vrcholy jsou tvořeny záznamy a vztahy mezi nimi jsou znázorněny pomocí hran. Samotné záznamy se mohou nacházet v jedné nebo ve vícero z následujících rolí: nadřízený (rodič), podřízený (potomek) a sourozenec. Vazby mezi položkami definují strukturu stromu a jeho vlastnosti. Názorná ukázka pro ujasnění jednotlivých vazeb je převzata z RFC 4512 [21]. Existuje-li hrana z bodu X do Y, potom je záznam v X bezprostředním předchůdcem Y a Y je zároveň potomek X jak ukazuje obrázek 2.2.



Obrázek 2.2: Vazby mezi záznamy v DIT z RFC 4512 [21].

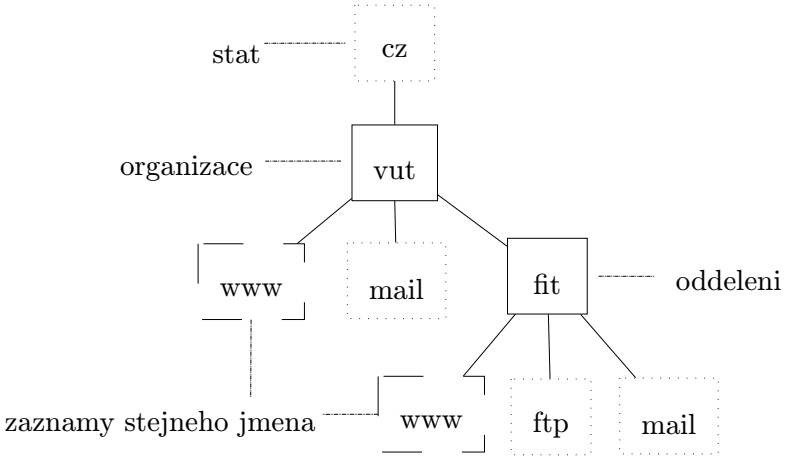
Samotná struktura adresářového informačního stromu se nejčastěji tvoří podle hierarchie organizace. Vezme-li se posloupnost od vrcholu směrem dolů, je běžné použití následujícího pořadí: zkratka státu, jméno organizace, oddělení atd. Tento příklad je zobrazen na obrázku 2.3. Tímto se dostáváme k důležité vlastnosti, která vyplývá z určité struktury stromu, jde o jmenný prostor.

Jmenný prostor je tvořen z jednotlivých názvů záznamů. Takzvaný kořen stromu definuje vstupní položku pro jednotlivé operace nad LDAP (více v části funkční model 2.3, která se podrobněji zabývá problematikou operací nad LDAP) a je převážně definován při prvotním nastavení LDAP. Kořen je identifikován jménem serveru a číslem portu, na kterém adresářová služba běží. Pod adresářovým kořenem se nachází adresářový sufix, který obvykle prezentuje organizaci nebo její doménové jméno. Název každého záznamu je ve vztahu s jeho umístěním. V jedné úrovni se svými sourozenci musí být pojmenování jedinečné, ale v širším pohledu na strukturu se již jména mohou opakovat. Tento příklad názorně ukazuje obrázek 2.3, který je s mírnou modifikací převzat z [9] ze strany 57. Hlavním cílem obrázku je ukázka použití stejného pojmenování v hierarchickém jmenném prostoru. Jak je patrné, záznam se jménem *www* je v podstromu *vut*. Také jiná položka se stejným jménem *www* se může nacházet ve stromu, ale nemůže již být sourozencem dříve uvedeného záznamu. Rozdíl mezi těmito položkami je v jejich rodičích *vut* a *fit*.

Hierarchie LDAP

Podíváme-li se na strukturu LDAP z pohledu souborového systému, zjistíme, že se zde nachází několik rozdílů. Následující tři uvedené body zachycují diferenci mezi těmito systémy z pohledu pana Howese [12]. Rozdíly jsou následující:

1. Souborové systémy jsou postaveny na jedinečném adresáři nazvaném kořen. LDAP taktéž má adresář nazvaný kořen, ale je jiný než v souborovém systému. Jeho hlavním



Obrázek 2.3: Ukázka použití stejného pojmenování v hierarchickém jmenném prostoru.

cílem je vytvoření tzv. vstupní brány do adresářové služby a poskytnutí základních informací o LDAP systému.

2. Souborové systémy rozlišují pojem adresář a soubor. Jednotka nemůže být zároveň adresář a soubor. Pouze adresář může obsahovat potomka. Namísto struktura u LDAP tyto možnosti nabízí. Každý uzel ze stromové struktury muže být současně jak kontejner (obsahuje podadresáře), tak může obsahovat data.

3. Rozdíl mezi pojmenováním uzlů LDAP a souborového systému. Při čtení cesty k adresáři nebo k souboru zprava doleva se ve stromové struktuře pohybuje od kořenu k danému souboru. U LDAP je to přesně naopak, tedy při pohybu od kořene k danému uzlu je nutné číst zleva doprava.

2.2.2 DN a RDN

U adresářového informačního stromu jsou všechny uzly pojmenované pomocí tzv. relativně rozlišovacího jména neboli RDN (Relative Distinguished Name). RDN definuje způsob, jakým jsou jednotlivé položky a data jednoznačně určeny v DIT. Je vybrán jeden nebo více atributů, které slouží k identifikaci položky. Hodnoty atributu musí splňovat následující podmínku:

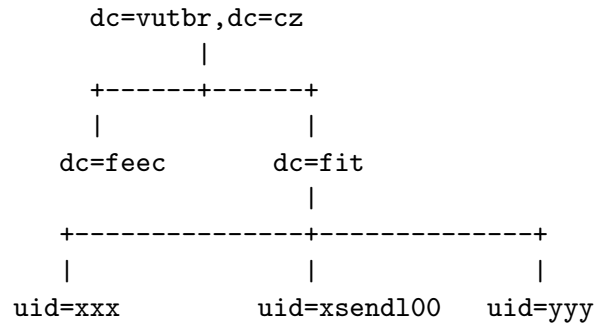
- jedinečnost mezi sourozenci v dané výšce podstromu.

Hodnoty RDN by měly být neměnné a neměly by obsahovat interní nebo privátní informace a to z důvodu jejich viditelnosti zvenčí.

Při operacích (popsány v části o funkčním modelu 2.3), které modifikují adresář a nebo při autentizaci, je využit plně kvalifikovaný název tzv. DN (Distinguished Name). Dle RFC 4512 [21] je DN složen z RDN a z DN, jeho přímého nadřízeného. Tento pojem jednoznačně kvalifikuje záznam v celém stromě. Tak jako je budovám přidělena jednoznačná adresa, tak i záznamy obsahují názvy (DN), pomocí nichž je možné je rozlišovat.

Obrázek 2.4 zobrazuje jednoduchý strom, na kterém je předvedena tvorba jednoznačného jména. Vezme-li se záznam s RDN *uid=xsendl00* a půjde-li se směrem nahoru ke kořenu, tak se potupně získá DN. Otec našeho záznamu má RND *dc=fit* a jeho otec neboli

kořen je *dc=vutbr,dc=cz*. Tedy postupným procházením stromu zdola nahoru se získá úplné jednoznačné jméno: *uid=xsendl00, dc=fit, dc=vubr, dc=cz*.



Obrázek 2.4: Tvorba jednoznačného jména (DN).

Relativně rozlišovací jméno lze chápat jako rozlišovací jméno položky v rámci adresářové struktury, něco jako primární klíč v tabulce v relačních databázích. Namísto toho jednoznačné jméno je něco jako spojení názvu tabulky s privátním klíčem.

2.3 Funkční model

Po popisu modelu, který definoval způsob uložení adresářů ve stromové struktuře, nyní přichází na řadu model funkční. Funkční model popisuje operace, které lze využít na adresářích a protokolu LDAP. Jak uvádí Howes [12], model je složen z množiny operací, které jsou rozděleny do tří základních skupin. Jejich výčet je následující:

1. dotazovací operace,
2. aktualizační operace,
3. autentizace a kontrolní operace.

Od verze protokolu LDAPv3 je možné tuto sadu operací rozšířit pomocí freamworku (LDAP extended operations), aniž by byla nutná další úprava protokolu. Nyní budou rozebrány jednotlivé skupiny operací podrobněji a na jednoduchých příkladech bude nastíněno jejich použití.

2.3.1 Dotazovací operace

Dotazovací operace umožňuje klientovi číst data ze serveru. U protokolu LDAP neexistuje žádná operace typu: „přečti hodnotu atribut“. Tato činnost je zde suplována pomocí vyhledávací funkce a následném předání informací zpět ze serveru na klienta. Výsledek vyhledávání je ovlivněn několika aspekty, ať už to jsou parametry nastavení vyhledávací funkce nebo nastavení práv v adresářovém stromu. Operace *search* má několik povinných atributů [21]. Jejich výčet je následující:

1. **základní objekt** - objekt, ve kterém začne vyhledávání,
2. **rozsah** - určuje rozsah vyhledávání (3 typy),
3. **dereference aliasů** ,

4. **velikostní limit** - maximální počet záznamů, které server nalezl jako splňující vyhledávací podmínky,
5. **časový limit** - maximální čas pro vyhledávání (v sekundách),
6. **atributy nebo hodnoty** - indikátor, zda se do výsledku vyhledávání zahrnují i hodnoty atributů nebo pouze atributy,
7. **filtr** - slouží k určení podmínky, pouze záznamy splňující podmínku budou vráceny (viz níže),
8. **atributy** - seznam atributů, které budou vráceny.

Přesná definice vyhledávacího požadavku je k nalezení v dokumentu RFC 4511 [17]. Nyní budou rozebrány části, které jsou pro tuto práci nezbytné.

Vyhledávací parametry

Pomocí základního objektu se definuje místo ve stromě, které je možné chápat jako startovací bod pro vyhledávání. Parametrem rozsah lze definovat velikost oblasti, která bude využita při operaci *search*. Tedy vyčleňuje část stromu, ve kterém se bude hledat shoda se zadanými požadavky. Existují tři typy rozsahu. Prvním je vyhledávání pouze v objektu, který je definován jako základní. To znamená, že se bude hledat shoda pouze na jediném místě, žádná jiná část stromu nebude procházena. Druhý typ již nabízí větší část stromu, jde konkrétně o všechny první syny základního objektu. Poslední možnost, tzv. „celý podstrom“, definuje největší možnou část pro prohledávání. Jsou to všechny podřízené položky základního objektu.

Základní objekt a rozsah vyčlení oblast pro hledání, kde budou porovnány všechny atributy. K hledání a navrácení výsledků pouze určitých záznamů slouží tzv. filtry. Pomocí nich lze vybrat pouze požadované hodnoty. Existuje řada filtrů jako je vyhledávání podřetězců, řadící shody, shoda rovnosti a další.

2.3.2 Autentizace a kontrolní operace

První operací, která se provede při snaze získat informace z LDAP serveru je navázání spojení (bind). Funkce umožní ověřit informace vyměňované mezi klientem a serverem, je zde chápána jako autentizace.

2.3.3 Aktualizační operace

Adresářová služba není navržena pro časté změny dat, tak jako databáze. Přesto je nezbytná existence operací, které dokáží s adresáři pracovat i jinak než v nich pouze vyhledávat a hledat shodu se vzorem. Operace, které mají schopnost upravovat strukturu dat, patří do skupiny aktualizační. Do této kategorie se řadí: přidávání, mazání a přepisování záznamu.

Pro přidání záznamu do adresáře na serveru použije klient operaci *add*. Zpráva pro server je složena ze dvou částí, kde první díl (DN) definuje jednoznačné jméno položky, která má být přidána na server. Díky stromové struktuře je tento údaj, který splňuje nezbytné podmínky (viz RFC 4511 [17]), dostačující pro jednoznačné umístění ve stromové struktuře. Druhá část zprávy pro server může být tvořena seznamem dvojic: atribut a data. Nebo-li, pokud objekt obsahuje některé povinné atributy, je nutné, aby byly nastaveny.

Po přidání záznamu na serveru přichází na řadu operace *delete*, která zprostředkovává klientovi možnost smazat položku z adresáře. Obsahuje pouze jediný povinný atribut a to jednoznačné jméno objektu, který se má odstranit. Po přijetí zprávy, server musí rozhodnout, zdali je možné záznam ze stromu odstranit. Pouze uzel, který nevlastní potomky (ukončuje danou větev stromu), je možné touto operací bezpečně smazat. V jiném případě je operace serverem odmítnuta a klient obdrží odpověď, prostřednictvím níž se dozví informace o nezdařilé operaci.

Změna DN umožňuje klientovi měnit RDN záznamu v adresáři. Se změnou uzlu RDN je spjat i jeho celý podstrom. To znamená, že změní-li se RDN uzlu, který není koncový, zároveň dojde k přemístění všech jeho potomků na novou pozici. Požadavek modifikace zaslaný na server má následující čtyři parametry: jednoznačné jméno záznamu (který se bude měnit), nové jméno záznamu (RDN), hodnota zdali se má staré RDN záznamu změnit na atribut nebo smazat a nakonec název nového rodiče modifikovaného záznamu.

Operace porovnání umožňuje klientovi porovnat hodnotu výrazu s hodnotami určitého atributu záznamu v adresáři. Parametry jsou: jednoznačné jméno záznamu, který bude porovnáván a hodnota atributu. Touto operací lze jednoduše zjistit, zda daný záznam obsahuje požadovaný parametr, v kladném případě server odpoví pozitivně, jinak vrací chybu nebo negativní odpověď.

Použije-li klient některou z výše uvedených operací a z nějakého důvodu ji bude chtít zrušit, využije k tomuto kroku operaci *abandon*. Všechny činnosti serveru zadané klientem ale ukončit nelze. Patří k nim operace *bind*, *unbind*, *abandon* a *StartTls*.

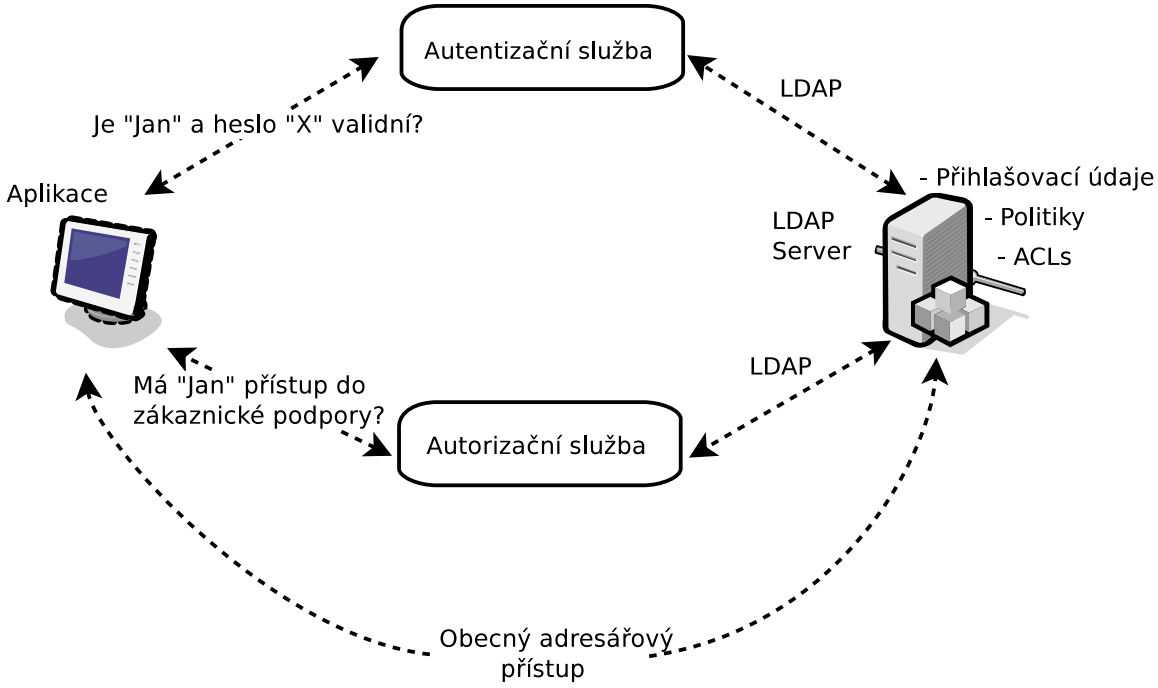
2.4 Bezpečnostní model

Poslední model, který zde bude rozebrán, se nazývá model bezpečnostní. Model je převážně využíván k autentizaci a autorizaci klienta (uživatele) na serveru, k zajištění soukromí, dostupnosti a integrity dat. Žádná z těchto funkcí není přímo nabízena LDAP serverem, ten pouze poskytuje prostředky k jejich zajištění. Jak je patrné z obrázku 2.5, kde je znázorněn průběh výměny informací mezi aplikacemi, autorizačními a autentizačními službami s LDAP serverem.

V době, kdy se každodenně objevují zprávy o krádeži identity nebo citlivých dat, je nutné bezpečnosti věnovat velkou pozornost. Za tímto účelem je intenzivně doporučeno využívat nejmodernější bezpečnostní prvky, které jsou nabízeny. Jelikož je při komunikaci mezi klientem a serverem nejčastěji jako komunikační kanál využíván internet, hrozba odposlechu třetí stanou je velká. Přidá-li se pro přenos surových dat využití takzvaného nezabezpečeného kanálu, vytváří se tím pro útočníky téměř ideální prostředí pro odchyt dat. Tato situace však u LDAP serverů nastává pouze zřídka a to z důvodu, že právě LDAP server je převážně využíván jako vstupní brána k dalším systémům. Za tímto účelem nabízí nespočet bezpečnostních možností. V následujícím textu budou nastíněny pouze ty možnosti, které jsou v diplomové práci využity. Lze je rozdělit na část zabývající se přenosem dat mezi klientem a serverem a na část, která předurčuje vlastnosti uživatelů a tedy definuje chování serveru.

2.4.1 Zabezpečený kanál

Při snaze klienta o komunikaci se serverem je nutné, aby znal adresu a číslo portu, na kterém server naslouchá. Již v této části je možné provést určité bezpečnostní opatření a to využití



Obrázek 2.5: Výměna informací mezi LDAP serverem a externími službami [9].

portu 636 a SSL/TLS. V části návrhu 4.1, zabývající se bezpečností, je ale doporučováno využití standardního portu 389 s kombinací se StartTLS.

Samotnou komunikaci se serverem je možné provozovat ve dvou režimech: zabezpečený a veřejný. V prvním případě je pro přenos dat využito zašifrovaného kanálu a pro navázání spojení jsou nutné přihlašovací údaje v podobě jména a hesla. Po úspěšném přihlášení je uživatel oprávněn ke CRUD[5] operacím. Na druhou stranu nezabezpečená komunikace, při které není uživatelova totožnost ověřována, převážně slouží pouze ke čtení dat.

2.4.2 ACL

Seznam řízeného přístupu neboli Access Control Lists (ACL) je možnost, kterou lze určit chování serveru. Jde o seznam jednoduchých příkazů, při kterých je nadefinováno *kdo* a *jakým způsobem* má přístup k jednotlivým *částem LDAP* struktury [7, 8]. Pro zápis politik je využita jednoduchá, ale flexibilní syntaxe. Gramatika používaná u systému OpenLDAP je následující:

```
access to [resources]                                    (1)
       by [who] [type of access granted] [control]       (2)
```

Obrázek 2.6: Struktura pravidla (ACL) [7].

První řádek (1) specifikuje k jaké entitě v adresářovém stromě je celé pravidlo vztaženo. Definuje to co má být omezeno a to prostřednictvím DN, atributu, filtru nebo jejich

[5] create, read, update, delete - vytvořit, číst, aktualizovat, smazat

kombinací. Druhý řádek gramatiky pravidla ACL uvedeném na obrázku 2.6 je složen ze tří následujících částí:

Kdo - neboli *who* je položka indikující, jaké entity mají povolen přístup ke zdrojům identifikovaným v částí *access*.

Typ přístupu - neboli *type of access granted*, indikuje co je možné provádět se zdroji.

Řídící pole - neboli *control*.

Klíčové slovo *by* na začátku 2. řádku určuje, ke kterým objektům je pravidlo vztaženo. Hodnota tohoto parametru je velice rozmanitá, poskytuje přes dvacet forem, které lze následně společně kombinovat. Přesto existuje pět nejčastěji využívaných typů, ty jsou zobrazeny v tabulce 2.4, která je tvořena hodnotou a jejím popisem.

Hodnota	**Popis hodnoty**
*	Globální shoda zahrnuje veškeré klienty, včetně anonymního připojení.
anonymous	Neautentizované připojení, připojení pomocí anonymního uživatele.
self	Obsahuje doménové jméno aktuálně připojeného uživatele, který byl úspěšně autentizován.
user	Autentizované připojení.
dn	Doménové jméno určeno pomocí regulárního výrazu.

Tabulka 2.4: Výčet nejčastějších hodnot definujících *kdo* má přístup ke zdroji.

Pro definování oprávnění je k dispozici šest následujících typů a jeden typ pro neudělení žádného oprávnění:

zápis - neboli *write* (w) uděluje oprávnění k zápisu záznamu nebo atributu,

čtení - neboli *read* (r) oprávnění pro čtení záznamu nebo atributu,

vyhledávání - neboli *search* (s) oprávnění pro vyhledávání záznamů a atributů,

porovnání - neboli *compare* (c) oprávnění použití porovnávacích operací na atributy nebo záznamy,

autentizace - neboli *auth* (x) oprávnění pro autentizaci,

existence - neboli *disclose* (d) oprávnění pro zjištění, zda záznam nebo atribut existuje,

žádné - neboli *-wrscxd* (0) zamítnutí všech oprávnění.

Obrázek 2.7, který zobrazuje komplexní příklad [8], představuje sílu a přehlednost zápisu ACL. Pravidlo definuje chování serveru na entity umístěné v *ou=eng,dc=plainjoe,dc=org* (1) a v jeho podstromě. Konkrétně jde o entity reprezentující uživatele a jedná se o parametr *userPassword* (2). Přihlášený uživatel má možnost si své heslo změnit (3). Na druhou stranu všichni ostatní atribut s heslem mohou použít pouze pro autentizaci (4). Poslední řádek (5) pravidla určuje podskupině *ou=admins,ou=eng,dc=plainjoe,dc=org* stejná oprávnění jako vlastníkovi (jde o administrátory).

```
access to dn.chilren="ou=eng,dc=plainjoe,dc=org"                    (1)
   attrs=userPassword                                               (2)
   by self write                                                    (3)
   by * auth                                                        (4)
   by dn.children="ou=admins,ou=eng,dc=plainjoe,dc=org" write       (5)
```

Obrázek 2.7: Ukázka komplexního pravidal ACL [8].

Kapitola 3

Požadavky a analýza

Třetí kapitola se zabývá procesem analýzy a specifikací požadavků. Stanovuje služby, které by měla aplikace nabízet a vymezuje podmínky, za kterých je možné tyto funkce provozovat a vyžadovat.

Začátek kapitoly je zaměřen na prozkoumání již existujících řešení, nabízející totožnou nebo alespoň podobnou funkcionalitu. Při hledání aplikací bylo převážně využito obchodu Google Play[1], který patří k největšímu shromaždišti aplikací pro Android. Druhá část nazvaná analýza, se snaží vystihnout základní body, jež by měly být aplikací splněny. Poslední část se zaměřuje na požadavky, které jsou rozděleny na dvě části: funkční požadavky a nefunkční požadavky.

3.1 Současné systémy

Pro hledání současných řešení nabízející zálohování kontaktů, byla použita následující kritéria:

1. Aplikace musí být určena pro mobilní zařízení.
2. Podporovaný systém musí být Android.
3. Aplikace musí být šiřitelná zdarma.
4. Možnost využití vlastního serveru.
5. Použití LDAP.
6. Synchronizace.

Vyhledávání bylo prováděno na stránkách Google Play, které jsou určeny operačnímu systému Android a slouží jako vstupní brána pro získání aplikací oficiální cestou podporovanou firmou Google. Jiné zdroje pro nalezení současných systémů umožňujících synchronizaci kontaktů nebyly využity z důvodu rozsáhlosti tohoto obchodu a počtu obsažených aplikací.

U prvních dvou kritérií je zřejmé, z jakých důvodů byly při hledání stávajících řešení vyžadovány. Třetí bod (šíření zdarma) byl zahrnut za účelem možnosti prověření funkčnosti aplikace. V opačném případě je nutné za aplikaci zaplatit, i když je tato částka převážně symbolická, nepovažuji tyto aplikace za konkurenci. Ke zpoplatněným aplikacím lze jako

[1] https://play.google.com/

příklad uvést *LDAP Sync*. Z důvodu existence pouze placené verze, hodnocení a funkčnost aplikace vychází z komentářů uživatelů na stránce aplikace, kde je možné ji zakoupit. Většina uživatelů měla s touto aplikací nemalé problémy a vyčítali ji například nemožnost editace vytvořeného spojení se serverem. K základním problémům, díky nimž není možné tuto aplikaci považovat za konkurenci, je nepředvídatelná funkčnost synchronizace. Dle komentářů se provede pouze napoprvé a dále buď probíhá pouze směrem server →klient nebo vůbec.

Souhrn základních informací je zobrazen v tabulce 3.1, ze které je z údajů ohledně počtu stažení možné vyčíst, že o tuto aplikaci velký zájem není. To se dá přisoudit pouze existenci placené verze a k negativním komentářům, které převážně hodnotí záporně právě synchronizaci.

Kategorie	**Popis**
název	LDAP Sync
cena	placená (kolem 25,-Kč)
poslední verze	29. 07. 2011, 1.5
počet stažení	přes 100
hodnocení	3,7/5 (39 hodnocení)
knihovna	UnboundID

Tabulka 3.1: Souhrn informací o aplikaci *LDAP Sync* ze dne 27. května 2014.

Čtvrtému a pátému bodu z požadovaných kritérií, které se týkají typu a možnosti výběru serveru, vyhovuje aplikace *LDAP Client*. Přesto i tato aplikace nesplňuje všechny podmínky, jelikož nabízí pouze vyhledávání na serveru a případné uložení nalezených kontaktů do mobilního telefonu. Šestý bod umožňující synchronizaci zde splněn není. K základním funkčnostem aplikace patří možnost vyhledat kontakty a to dle následujících atributů: jména, příjmení, celého jména, emailu, ID. Rozšíření k těmto atributům je nabídnuta položka *vlastní vyhledávání*, kde je nutné zadat ve správném tvaru jméno atributů a hledanou hodnotu. Tento přístup není pro uživatele, kteří s LDAP protokolem nikdy nepracovali, jednoduchý. Nalezené údaje je možné přidat do kontaktů, kde nastává problém, že jsou importovány pouze některé údaje a není nabídnuta žádná možnost mapování atributů na položky v kontaktech systému Android. Při zobrazení záznamu je možné na jednotlivé položky kliknout a provádět nad nimi určité operace v závislosti na typu hodnoty: poslat email, započít telefonní hovor, navigovat na danou adresu, zobrazit adresu a kopírovat obsah buňky. Základní informace ohledně ceny, vývoje a hodnocení je možné nalézt v tabulce 3.2.

3.2 Analýza

Cílem této práce je poskytnutí nástroje k nahrazení stávajících možností zálohování kontaktů v mobilních zařízení. Aplikace je určena především uživatelům, kteří nechtějí svá data poskytovat třetí straně a o bezpečnost a správu svých informací se chtějí starat sami.

Základním cílem systému je vytvoření řešení zálohování kontaktů na základě vlastních prostředků. Řešení se skládá ze samotné mobilní aplikace a adresářové služby. Privátně je aplikace navržena pro synchronizaci kontaktů, tedy údajů o různých uživatelích jako

Kategorie	Popis
název	LDAP Client
cena	zdarma
poslední verze	09. 09. 2010, 1.1
počet stažení	přes 10 000
hodnocení	3,8/5 (78 hodnocení)
bezpečnost	žádná, SSL, StartTLS
knihovna	UnboundID
zápory	- chybí nápověda
	- neumí vyhledávat podřetězce
	- nemožnost synchronizace přidaných kontaktů

Tabulka 3.2: Souhrn informací o aplikaci *LDAP client* ze dne 27. května 2014.

je jméno, příjmení, telefon, email a další. Aplikace je připravena pro operační systém Android, zejména z důvodu jeho velké rozšířenosti mezi uživateli. Synchronizace bude zajištěna pomocí protokolu LDAP. Pro serverovou část bude použita open sourcová implementace LDAP protokolu, software OpenLDAP. S využitím této aplikace, tedy nainstalováním ji na přenosné zařízení a nastavení LDAP serveru, dostane uživatel plnohodnotnou službu pro synchronizaci kontaktů v privátním cloudu ve své vlastní režii.

3.2.1 Základní poznatky

- Aplikace pracuje v módu klient server.
- Komunikace mezi klientem a serverem probíhá pomocí internetového připojení.
- Jako klient je zvoleno mobilní zařízení.
- Operační systém chytrého telefonu je Android.
- Grafické uživatelské rozhraní na mobilním telefonu využívá funkce Androidu verze 4.0.
- Server poskytuje adresářové služby (LDAPv3).
- Server obsluhuje a vyřizuje požadavky z mobilního zařízení.
- Mobilní telefon zobrazuje informace ze serveru a řídí synchronizaci.
- Pro dočasné informace je využita SQLite databáze na klientovi.

3.2.2 Proveditelnost na Android zařízení

Následující výčet nabízí přehled požadavků na mobilní aplikaci:

- zabezpečení interních informací,
- možnost propojení a komunikace s LDAP serverem,

- získání přístupu ke kontaktům a možnost jejich modifikace,
- získání přístupu k účtům a možnost jejich modifikace,
- získání přístupu ke čtení a zápisu na externím úložišti,
- získání přístupu ke čtení synchronizačních informací,
- využití internetového připojení,
- přístup k datovému nastavení.

3.3 Funkční a nefunkční požadavky

Po pečlivém přezkoumání cíle projektu a otázek na úrovni návrhu, vznikl seznam náležitostí, který by vyvíjená aplikace měla splňovat. Požadavky jsou v zásadě rozděleny do dvou následujících kategorií: funkční požadavky a nefunkční požadavky. Funkční požadavky jsou takové, které definují specifické funkce nebo chování aplikace. Funkční požadavky zachycují předpokládané chování systému. Toto chování může být vyjádřeno pomocí služeb, úkolů a funkcí systému. Při vývoji produktů, je vhodné rozlišovat mezi funkcemi nepostradatelnými, které musí splňovat všechny aplikace podobného typu, a vlastnostmi, odlišující systém od konkurence. Naproti tomu nefunkční požadavky jsou ty, které určují kritéria, která mohou být použita k posouzení, jak aplikace pracuje na rozdíl od jeho specifického chování. Obě kategorie jsou uvedeny níže.

3.3.1 Funkční požadavky

Dle slovníku terminologie softwarového inženýrství funkční požadavky specifikují funkce, které systém nebo systémové komponenty musí být schopny provádět [5]. Níže zobrazený seznam definuje základní funkční požadavky na mobilní aplikaci pracující na systému Android. Jsou to:

1. CRUD operace na serveru,
2. přihlášení na server,
3. test připojení,
4. nastavení zabezpečení aplikace,
5. WiFi/mobilní synchronizace,
6. nastavení času synchronizace,
7. manuální synchronizace,
8. zobrazení kontaktů,
9. úprava synchronizovaných kontaktů,
10. výběr kontaktů, které se budou synchronizovat,
11. přidat kontakt do telefonu,

12. vyhledat kontakt na serveru,
13. nahrát kontakty ze serveru,
14. nastavení vzhledu aplikace.

3.3.2 Nefunkční požadavky

Nefunkční požadavky se netýkají chování systému, ale jsou zaměřeny převážně na jeho kvalitu. Následující výčet zobrazuje nejčastější oblasti nefunkčních požadavků vztažených k vyvíjenému systému:

použitelnost - uživatel musí být schopen využívat tuto aplikaci bez předchozích znalostí o uživatelském rozhraní. V případě práce se samotnou aplikací je částečná znalost problematiky, kterou se aplikace zabývá, nutná a to kvůli tomu, že nepatří do oblasti triviální. GUI musí umožnit uživateli pracovat s různými funkcemi, a to s největší možnou lehkostí. GUI by skrz celou aplikaci mělo být jednotné a nevymykat se standardům zavedeným v systému Android. Všechny nabízené funkce aplikace by měly být dostupné maximálně na čtyři kliknutí. V případě jakýchkoliv problémů či nejasností musí být uživatel schopen najít řešení v nápovědě nebo pomocí tooltipů umístěných na složitějších místech aplikace. Vyvolání nápovědy by mělo být dostupné z kterékoliv části aplikace.

spolehlivost - aplikace využívá ke své práci internetové připojení, ale není na něm přímo závislá. Jelikož synchronizace bude probíhat v předem daných intervalech nebo pouze na manuální vyžádání uživatele, je možné činnost aplikace provozovat i bez internetového připojení. V případě ztráty internetového připojení nebo jeho nedostupnosti bude aplikace používat poslední data načtena ze synchronizačního serveru k zobrazení informací o kontaktech. Také v případě nezdařilé synchronizace se aplikace uvede do stavu před jejím započetím a o synchronizaci se pokusí v dalším naplánovaném termínu. K pádům aplikace, by nemělo docházet častěji než jednou za den. V situaci, kdy se dostane do stavu, ve kterém nelze pokračovat, by se měla aplikace sama restartovat a obnovit svůj stav do stabilní podoby před chybou.

výkon - aplikace bude podporovat jednu instanci na telefonu. Poté, co uživatel zadá požadavek na vyhledávání kontaktů, by mu měla být odpověď doručena do 15 sekund. Tato podmínka je samozřejmě úzce spjata s typem a kvalitou internetového připojení mobilního zařízení. Délka trvání synchronizace je podmíněna počtem kontaktů, velikostí změn a množstvím dat, které je nutné přenést na server. Aplikace při procesu synchronizace bude optimalizována k tomu, aby prováděla co nejmenší počet nutných operací a to především v situaci, kdy je nutné řešit konflikty mezi daty na serveru a klientem. Je nutné využití více vláken, aby hlavní GUI vlákno bylo vždy připraveno reagovat na uživatelské podněty. Procesy, na jejichž dokončení a výstupech bude závislé hlavní vlákno, by mělo být možné bezkolizně předčasně ukončit.

podporovatelnost - aplikace by měla být kompatibilní se všemi budoucími změnami. Při výměně vzdáleného synchronizačního serveru by měla být zachována spolupráce za podmínek, že zůstane zachována verze protokolu a adresářové služby (LDAPv3). Uživatel by měl být schopen aktualizovat aplikaci na novou verzi, aniž by bylo nutné restartování zařízení, a aniž by došlo k jakékoliv kolizi ať už z pohledu kontaktů nebo nastavení.

realizace - za předpokladu, že mobilní zařízení obsahuje sytém Android minimální verze 4.0 a vyšší a podporuje možnost internetového připojení ať už pomocí WiFi nebo mobilních dat, lze tuto aplikaci na daném zařízení používat. Tedy není nutný další zásah do systému android.

rozhraní - aplikace komunikuje se serverem a pomocí LDAPv3 využívá jeho služeb. Data na vzdáleném serveru jsou uložena v adresářích, jejichž strukturu si každý uživatel vytvoří sám při inicializaci a nastavení serveru. Aplikace předpokládá použití schématu *GoogleContact*, který byl pro tento účel vytvořen (více o schématu v části zabývající se implementací 5.2.1). Informace z adresářového serveru jsou vyměňovány přes internet.

provoz - uživatel by měl být schopen prostřednictvím mobilní aplikace získat aktuální verzi kontaktů uložených na serveru za předpokladu, že má přístup k internetu, vytvořen účet a server je správně nakonfigurován a je v provozu.

Kapitola 4

Návrh

Po analýze a specifikaci požadavků následuje další etapa ve vývoji softwaru. Je jím návrh, který na základě předchozích částí dokumentů a za pomocí modelovacích nástrojů, navrhuje základní stavební bloky systému. Informace a poznatky týkající se bezpečnosti v systému Android byly čerpány převážně z publikace [18] a část o návrhu vychází z webových stránek určených pro vývojáře [2].

V předchozí kapitole bylo shrnuto mnoho cílů v podobě požadavků, které jsou kladeny na vyvíjený systém. Následující část se zaměřuje na vybrané požadavky podrobněji. Konkrétně jde o oblasti, které se zaobírají problematikou bezpečnosti:

1. autentizace,
2. šifrování.

Jde především o zajištění přenosu dat z mobilního zařízení k jiným stranám, zejména se to týká komunikace klientské aplikace se serverem. V tomto případě jsou využity některé kryptografické konstrukce, které jsou v dnešní době doporučeny k zajištění ochrany uživatele.

Zbývající části kapitoly jsou zaměřeny na odlišnou oblast než je bezpečnost. Kladou si za cíl vytvoření nebo alespoň nastínění návrhu kostry aplikace. Převážně se jedná o pokročilou funkci jako je synchronizace jen vybraných kontaktů, splňující danou podmínku, aby bylo možné rozlišit privátní a korporátní data.

4.1 Důvěrnost a ověřování

V situaci, kdy jsou data přenášena mimo zařízení, je nutné mít v povědomí dva bezpečnostní aspekty. Prvním aspektem je *autentizace*, při které je ověřováno, zda subjekt s nímž komunikace probíhá je ten, za kterého jej považujeme. Dodržení této podmínky je důležité z mnoha důvodů. V případě, kdy jsou soukromé informace o kontaktech posílány na server je nutné zajistit, že cílové zařízení, které obdrží data je opravdu náš server určený pro synchronizaci. A ne podvodný objekt, který se snaží odhalit důvěryhodné informace. Na tuto situaci je možné pohlédnout i z druhé strany, kdy jsou data stahována ze serveru.

Druhý aspekt, *důvěrnost*, je zaměřen na ochranu dat při přenosu mezi objekty. Výměna informací probíhá skrze nezabezpečené veřejné sítě a je tedy nutné třetím stranám znemožnit nebo alespoň zkomplikovat jejich čtení.

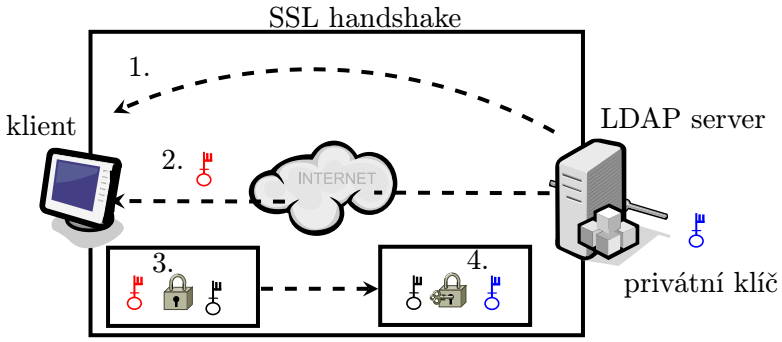
SSL/TLS

Pro výše uvedené aspekty existuje řešení v podobě SSL[1], které je určeno na ochranu dat při přenosu přes Internet. V roce 1999 vyšla jeho nová verze, která byla pojmenována TLS[2]. Rozdíly mezi verzí SSL a TLS jsou pouze nepatrné a v mnoha aplikací je možné najít oba standardy.

V úvodu kapitoly byly vyčleněny dva požadavky týkající se bezpečnosti sytému. Jde o *autentizace* a *šifrování*. Za účelem splnění těchto vybraných částí, bude při komunikaci mezi synchronizačním serverem a mobilním zařízením, využíván bezpečnostní kanál. Je založen na protokolu SSL/TLS, který je právě k těmto situacím navržen.

Zjednodušená komunikace mezi mobilním zařízením a serverem je zobrazena na obrázku 4.1. Během tří kroků si server s klientem vymění důležité informace jako je veřejný klíč, který patří serveru, a session klíč zašifrovaný pomocí veřejného klíče. Popis kroků je následující:

1. Klient pošle požadavek na server k zahájení zašifrované komunikace.
2. Server vlastní dva klíče, privátní (modrý) a veřejný (červený). Veřejný klíč zašle klientovi, který podle něj ověří zda je server ten za koho jej klient považuje.
3. Prostřednictvím veřejného klíče, získaného ze serveru, klient zašifruje náhodně vygenerovaný session klíč (černý) a zašle jej na server.
4. Server obdrží zprávu zašifrovanou veřejným klíčem. Pomocí svého privátního klíče ji rozkóduje a získá session klíč od klienta.



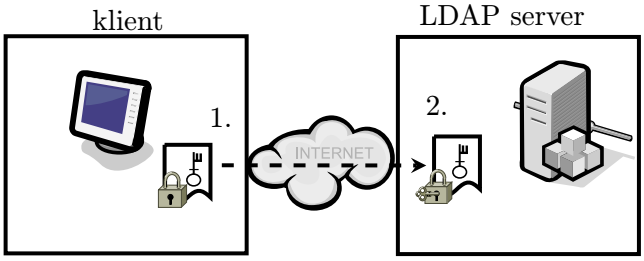
Obrázek 4.1: Handshake mezi klientem a serverem.

Uskuteční-li se výše zmíněné kroky bez jakýchkoliv chyb, jsou oba subjekty připraveny na výměnu informací šifrovaných prostřednictvím session klíče. Obrázkem 4.2 je ukázáno jakým způsobem je bezpečnostní proces výměny informací mezi serverem a klientem dokončen. Kroky jsou následující:

1. Klient zašifruje data session klíčem a odešle je na server.
2. Server data příjme a dešifruje je session klíčem.

[1] Secure Sockets Layer
[2] Transport Layer Security

Poslední dva kroky se odehrávají při veškeré výměně informací mezi oběma stranami, za předpokladu že je pro komunikaci využit kanál zabezpečený pomocí SSL/TLS. Tímto krokem by měla být data přenášená přes síť ochráněna před útočníky, jež bz chtěli komunikaci odposlouchávat.



Obrázek 4.2: Šifrování, dešifrování souborů.

Výše popsané bezpečnostních prvky budou využity z knihovny *UnboundID*, která nabízí rozhraní pro komunikaci se LDAP serverem.

StartTLS

Při vytváření nového připojení bude jako přednostně nabízeno spojení StartTLS, které za určitých podmínek využívá SSL/TLS. Tento typ komunikace je v dnešní době upřednostňován před samotným SSL/TLS. Technicky se tyto dvě bezpečnostní ochrany od sebe mnoho neliší. StartTLS nabízí možnost využití bezpečného přenosu dat v případě, že jej server poskytuje, v opačném případě jsou data přenášena nezabezpečeně. Druhým rozdílem, který souvisí s prvním, je použití stále stejného čísla portu jak pro nezabezpečený tak i pro zabezpečený přenos.

V nabídce při definování připojení k serveru na mobilním zařízení se bude nacházet i nabídka na využití nezabezpečeného přenosu, avšak uživatel bude důrazně upozorněn na možná bezpečnostní rizika.

4.2 Android

Pro tvorbu aplikací na operačním systému Android je využíváno tzv. Android SDK, které poskytuje knihovny API a vývojářské nástroje potřebné k sestavení, testování a ladění aplikací pro Android. Obsahem Androidu SDK je řada komponent z nichž jedna slouží pro správu centrálního úložiště na zařízeních. Jde o tzv. *contacts provider* (poskytovatel kontaktů). Jeho cílem je uchovávat data o lidech ve formě kontaktů a zpřístupňování těchto informací ostatním aplikacím. Další nezbytnou komponentou je tzv. *sync adapter*.

4.2.1 Synchronizační adaptér

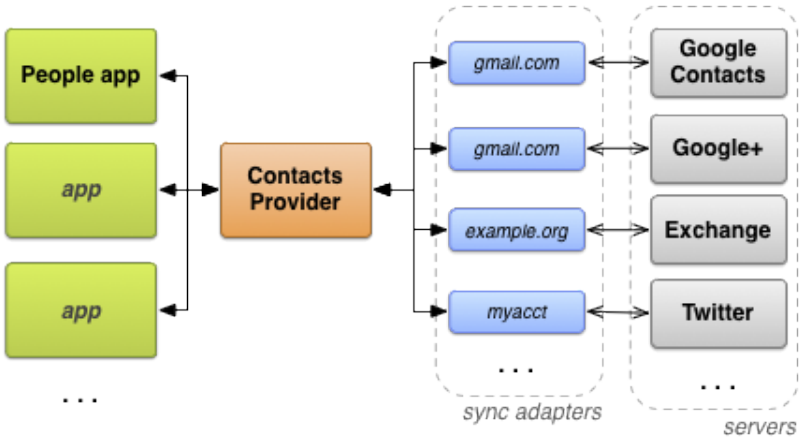
Jak je patrné z obrázku 4.3, systém Android poskytuje rozhraní pro synchronizaci kontaktů z mnoha účtů zároveň. Pro synchronizaci je tedy nutné implementovat synchronizační adaptér, který by využíval rozhraní umožňující přístup ke kontaktům, a který by řídil komunikaci a výměnu dat mezi LDAP serverem.

K hlavním výhodám využití tzv. synchronizačního adaptéru patří:

automatizace - při správné implementaci synchronizačního adaptéru odpadá starost o samotnou synchronizaci. Adaptér využije předem nadefinované funkce, které pouze v určitý okamžik spustí.

stav zařízení - synchronizace je provedena pouze za ideálních podmínek. Není tedy spuštěna v situaci, kde není k dispozici internetové připojení, nebo když je zařízení zaneprázdněno, nebo když mu to nedovolí stav baterie.

re-synchronizace - při neúspěšné synchronizaci, ať už z důvodu chyby serveru nebo připojení, je proces opětovně automaticky spuštěn ve vhodném okamžiku. Tento proces se automaticky opakuje do stavu, kdy je synchronizace provedena úspěšně.



Obrázek 4.3: Schéma synchronizačního adaptéru [2].

4.2.2 Poskytovatel kontaktů

Kontakt provider je výkonná a flexibilní Android komponenta, která slouží ke správě centrálního repositáře umístěného na daném zařízení. Poskytuje data, která je možné vidět v aplikacích zaměřených na práci s kontakty. Prostřednictvím jeho funkcí lze získat přístup k datům i pro vlastní aplikaci a přesunovat je mezi zařízením a online službami.

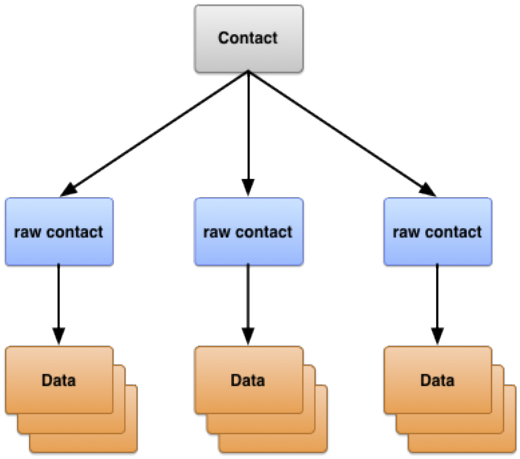
Kontakt provider [2] patří k nejdůležitějším komponentám v Androidu, které jsou v této práci využívány. Je schopen spravovat širokou škálu datových zdrojů s mnohými různorodými údaji a poskytovat rozsáhlé funkce pro práci s nimi.

Organizace dat

Při práci s osobami prostřednictvím providera kontaktů jsou využívány tři základní typy data. Každý tento typ přímo koresponduje s tabulkami v databázi. Struktura databáze a další podrobnosti budou rozebrány v dalších částech návrhu 4.3.1.

Obrázek 4.4, který je uveden níže, zobrazuje vztah mezi třemi základními typy zmíněnými výše. Typy jsou rozmístěny do tvaru stromu, kde je kořen tvořen objektem šedě podbarveným. Vrchol nese název *Contact* a jde o objekt, který zastřešuje všechny údaje o daném kontaktu z různých zdrojů. Tyto zdroje, podbarveně modře, jsou pojmenovány

raw_contact. V nejnižší vrstvě jsou zobrazena *Data*, která již uchovávají konkrétní údaje, jako je telefonní číslo, adresa a další.



Obrázek 4.4: Struktura tabulek poskytované kontakt providerem [2].

4.3 Návrh databáze

Aplikace vyvíjená v této práci je postavena na spolupráci s databází. Je uložena v adresářovém prostoru kontakt provideru pod názvem *contacts2.db* (na verzi Androidu 4.2.2 CyanogenMod)[3]. Jedná se o úložiště pro všechny informace, které se jsou soustředěny okolo kontaktů, hovorů a dalších souvisejících dat.

4.3.1 Databáze kontaktů

Kontakty v mobilním zařízení jsou uloženy v SQLite databázi, která se nachází na mobilním zařízení a obsahuje velké množství tabulek, pohledů, agregací, indexů a trigrů. V této práci jsou využity pouze následující tabulky:

Contacts - reprezentuje různé osoby. Tabulka je vytvořena prostřednictvím agregace z tabulky *raw_contact*.

Raw_contact - jedna entita představuje souhrnné kontaktní informace o osobě z jednoho uživatelského účtu a typu. Pro jednu osobu je možné mít více druhů záznamů. Ty se liší v názvu účtu a v jeho typu. Tato skutečnost umožňuje kombinovat údaje o jedné osobě z více zdrojů, ty jsou poté agregovány do tabulky *contacts*.

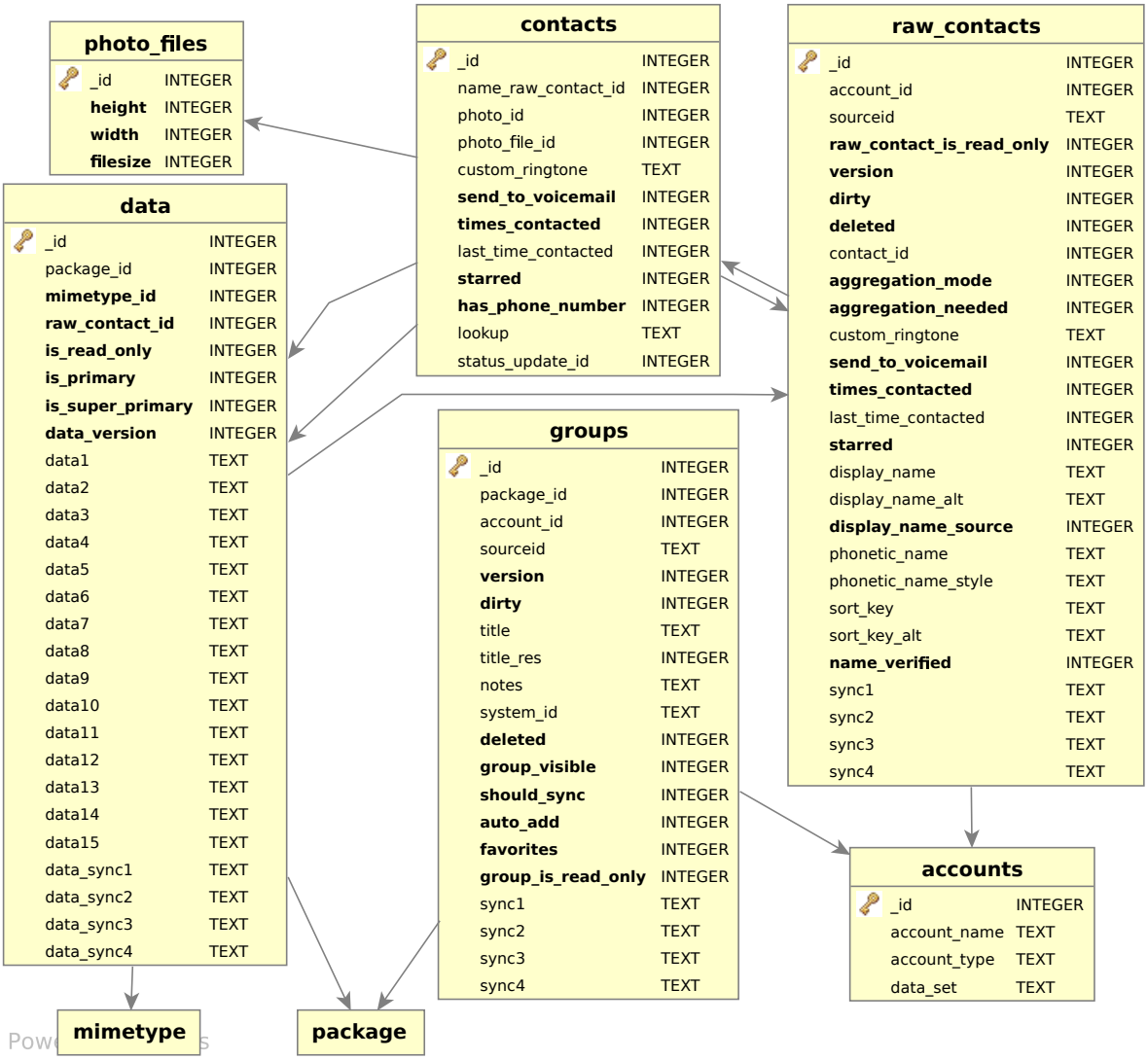
Data - detaily pro tabulku *raw_contact*.

Group - skupiny, do kterých je možné kontakty přiřadit.

Data, *raw_type* a *contacts* jsou stejné tabulky, zmíněné v části zabývající se kontakt providerem.

[3]Na jiných verzích Androidu se název databáze může lišit.

Obrázek 4.5 zobrazuje schéma vybraných tabulek a jejich závislostí. Je zde znázorněna pouze část databáze, která tvoří základní kostru pro kontakt provider. Obrázek je generovaný prostřednictvím aplikace *DbVisualizer*. Celé schéma databáze *contacts2.db* je uvedeno v příloze D.



Obrázek 4.5: Vybrané tabulky z databáze *contacts2.db*.

V části zabývající se implementací je vysvětleno jakou využitelnost mají tabulky a jejich vybrané položky, které jsou popsány v následujícím textu.

Účty

Kontakty na zařízení s operačním systémem Android jsou zařazeny do účtů. Jako příklad lze uvést Google, Skype, Facebook a další. V případě, že uživatel využívá synchronizaci a správu kontaktů nabízenou přímo Androidem, má všechny telefonní kontakty uloženy pod účtem Google. V situaci, kdy není využívána žádná služba a uživatel si pouze do

nového telefonu kontakty importoval, jsou kontakty uloženy v tzv. defaultním účtu. Ten se liší od výrobce k výrobci[4]. Informace o typu a názvu účtu, do něhož kontakty v zařízení patří, jsou uloženy v tabulce *accounts*. Pokud kontakt není zařazen do žádného účtu, je vytvořen nový záznam v tabulce *accounts*, jehož položky jsou nastaveny na `null`.

Raw_contact

Row_contact je tabulka reprezentující kontakty. Neobsahuje data, slouží pouze jako rodičovská tabulka pro další záznamy. ID entity slouží jako cizí klíč v tabulkách, ve kterých jsou již soustředěny konkrétní data. Díky těmto vlastnostem je možné pro jeden kontakt nadefinovat více hodnot pro stejný typ informací.

Pro samotnou synchronizaci hraje velkou roli sloupec *dirty*. Může nabývat hodnot *0* znázorňující, že kontakt nebyl modifikován od poslední synchronizace, nebo hodnotu *1* vyjadřující změnu. Hodnotu automaticky nastavuje kontakt provider a to v situaci, kdy Android aplikace aktualizují některou položku. Sloupec *deleted* je číselného typu a nabývá hodnot jako *dirty*. Jak již název napovídá jde o sloupec, jehož hodnoty definují zdali je kontakt smazán či ne. Pokud záznam nepatří do žádného účtu, je kontakt po zavolání operace na odstranění ihned smazán z databáze. V opačném případě operační systém očekává, že je záznam součástí synchronizačního procesu a pouze nastaví hodnotu *delete* na 1 a nadále zůstává v databázi. Pro úplné odstranění je nutné volat požadavek na smazání s parametry `CALLER_IS_SYNCADAPTER`.

Pro uchování metadat o synchronizovaných kontaktech jsou využity sloupce *sync1* až *sync4*. Sloupec *sync1* nabývá hodnot *0* a *1*. Záznamy označené 1 reprezentují kontakty vybrané uživatelem, které by se měly synchronizovat. Naopak s hodnotou 0 jsou ty, které nebudou aplikací *SyncContact* zálohovány. Pro určení zdali jsou na LDAP serveru novější data než na klientovi se využívá sloupec *sync2*, který nese časové razítko o poslední úspěšné synchronizaci. Poslední položka *uuid*, umístěna v *sync4*, je taktéž využita při komunikaci s cloudovým úložištěm. Jde o takzvané RDN (popsané v části zabývající se LDAP 2.2.2), pod kterým jsou kontakty uloženy na straně serveru.

Data

Konkrétní hodnoty pro kontakty jsou uloženy v tabulce *data*. Obsahuje dva cizí klíče *mimetype_id* a *raw_contact_id*. První odkazuje na tabulku *mimetype*, která definuje o jaký typ dat se jedná. Jako příklad lze uvést následující druhy: data reprezentující adresu, email, telefonní číslo, online komunikační sítě a další.

Druhý cizí klíč definuje vztah k tabulce *raw_contact* (popsána výše). Sloupce označené názvem *DataX* jsou určeny pro uchování informací. Pro každý typ definovaný v tabulce *mimetype* jsou hodnoty ve sloupcích *data* různé.

Z důvodu nutnosti převést kontakty, které se mají synchronizovat, na nově vytvořený účet je nezbytné zachovat si informace o účtu předešlém (více o nutnosti převádění kontaktů na nový účet v kapitole zabývající se implementací). Za tímto účelem byl vytvořen nový *mimetype* `vnd.android.cursor.item/cz.synccontact`. Pro metadata z tabulky *contacts* o předchozím účtu je vytvořen nový záznam v tabulce *data*. Hodnoty sloupců jsou následující: *Data1* – jméno předešlého účtu a *Data2* – typ předešlého účtu. Tyto údaje jsou využívány v situaci, kdy uživatel již nechce používat tento synchronizační nástroj a má v plánu vrátit své kontakty do původního stavu.

[4] Zařazení kontaktů v účtech se na různých verzích Android mohou lišit.

Při vytvoření nového kontaktu pod účtem této aplikace jsou hodnoty v záznamu nesoucí informace o předešlém účtu nastaveny na `null`. Při exportu kontaktů do předcházejícího účtu nejsou tyto záznamy uloženy pod žádným účtem. Jsou pouze označeny jako kontakty v mobilním telefonu.

Shodně jako tabulka *raw_contacts*, která uchovává informace o tom, jaké kontakty se synchronizují, tak i tabulka *groups* udržuje metadata o skupinách určených k synchronizaci. Význam jednotlivých *sync* sloupců je totožný. Informace o tom do jaké skupiny kontakt patří jsou standardně uloženy v tabulce *data* s položkou *mimetype* `GroupMembers`.

4.3.2 Synchronizace vybraných kontaktů

Pro návrh pokročilé funkce, která bude synchronizovat jen vybrané kontakty, se nabízí využití prostředků, které poskytuje SDK. Konkrétně se jedná o možnost členit kontakty do různých skupin na jejichž základě bude rozhodnuto, které kontakty budou podléhat synchronizaci. Uživateli využívající vyvíjený systém bude nabídnuta možnost manuálního zařazení kontaktů do skupin. Skupiny budou jak již existují (předefinované v Androidu nebo již dříve nadefinované uživatelem) nebo nově vytvořené v této aplikaci.

Synchronizované kontakty lze tedy vybrat přímo ze seznamu kontaktů, nebo prostřednictvím celé skupinu.

4.4 Synchronizace

Cílem práce je vytvoření aplikace pro mobilní zařízení Android, která by umožňovala synchronizaci kontaktů ve vlastním cloudovém úložišti. Jádrem celého tohoto projektu a také nejsložitější částí je proces synchronizace. Jedná se o skupinu funkcí, která bude volána prostřednictvím synchronizačního adaptéru Androidu nebo po uživatelově iniciativě.

Synchronizaci kontaktů, která by měla být touto prací implementována, lze považovat ze obousměrnou. Jelikož se skládá nejméně ze dvou uzlů (centrální server a jedno nebo více zařízení postavené na Androidu) lze ji klasifikovat jako distribuovaný systém. Jsou i splněny další podmínky, jako je přístup ke stejným údajům (seznam kontaktů) nebo snaha dosáhnout účinného šíření kontaktních informací a změn prostřednictvím zasílání zpráv v síti. I když jako centrální server byl vybrán systém adresářové struktury, jež lze považovat za statický.

4.4.1 Výběr algoritmu

Následující část se zaměřuje na výběr algoritmu, který bude synchronizační službou implementován [15].

V případě, kdy synchronizace v tomto projektu byla klasifikována jako distribuovaný systém, je nutné se zmínit, že tento fakt přináší omezení v podobě tzv. CAP teorému. CAP teorém[5] (konzistence, dostupnost a tolerance rozdělení) byl v roce 2000 představen Ericem A. Brewerem. Hlavní myšlenku lze prezentovat, následovně: *„není možné dosáhnout všech tří požadavků zároveň: dostupnost, konzistence a tolerance rozdělení“*. Z tohoto důvodu bude nutné v práci provést kompromis v podobě omezené konzistence. Konzistence je vlastnost zaručující, že všechny uzly v systému vlastní stejná data v každém okamžiku. Je patrné, že tento stav není možné zaručit. Naproti tomu podmínku dostupnosti a toleranci k rozdělení je možné splnit.

[5] Consistent, Available, Partition-Tolerant.

Při výběru algoritmu byla inspirace hledána ve standardu *SyncML*[6], který obsahuje sedm návrhů algoritmu pro synchronizaci. Pouze dva jsou založeny na obousměrné synchronizaci a iniciátorem je vždy klient (two-way sync a slow two-way sync). Tyto algoritmy by byly vhodné pro tuto diplomovou práci. Avšak po podrobnějším rozebrání bylo zjištěno, že je nutná činnost serveru. Konkrétně jde o to, že klient pošle změny na server, ten je zpracuje a pošle výsledek spolu se svou sadou změn zpět klientovi. Jak je patrné je nutné mít u tohoto procesu specializovaný server, za což se LDAP server považovat nedá.

4.4.2 Algoritmus

Pro synchronizaci byl použit algoritmus z práce [15], který prošel pouze úpravami pro potřeby tohoto systému.

Vystupují zde čtyři následující množiny:

$C_{local}^{conflict}$ - lokální kontakty, které se mají synchronizovat, jsou změněny od poslední synchronizace a jsou v konfliktu s kontakty na serveru.

C_{local} - lokální kontakty, které se mají synchronizovat a jsou změněny od poslední synchronizace

$C_{server}^{conflict}$ - kontakty uložené na serveru, které se mají synchronizovat, jsou změněny od poslední synchronizace a jsou v konfliktu se změněnými kontakty na zařízení.

C_{server} - kontakty uložené na serveru, které se mají synchronizovat a jsou změněny od poslední synchronizace.

Výše popsané množiny jsou základními objekty vystupující v algoritmu 4.1. V první fázi (1,2) jsou získány všechny kontakty ze serveru i z lokální databáze, které jsou od poslední synchronizace modifikovány a v systému jsou označeny, že mají být synchronizovány. Krokem 3. je vytvořena množina obsahující pouze kontakty, které jsou v konfliktu. Byly modifikovány jak na serveru tak i lokálně. 4. a 5. krok vytvoří množiny „bezproblémových“ kontaktů.

Cyklus `for` postupně prochází množinu kontaktů v konfliktu. Od kroku 11. až po krok 15. jsou řešeny lokální a LDAP konflikty u kontaktů. Jak je zřejmé z řádku 12, hlavní roli zde hraje čas modifikace. Novější úpravy přepisují úpravy starší.

V případě (17), kdy jsou kontakty stejné (hodnoty jejich atributů nejsou rozdílné) jsou po jednom přidány do bezkonfliktních množin.

Cyklus končí v případě, kdy jsou porovnány poslední modifikované kontakty. Před uložením a odesláním kontaktů na server jsou již pouze sloučeny množiny „nekolizní“ s množinami „kolizními“ (22, 23).

Ve zbývající části jsou data odeslána na server a lokálně uložena. Dále je nastaveno nové časové razítko poslední úspěšné synchronizace a vymazány „dirty“ položky u kontaktů.

4.4.3 Sekvence funkcí při synchronizaci

Všechny výše uvedené situace jsou demonstrovány na obrázku . Sekvenční diagram 4.6 obsahuje šest základních účastníků interakce, kteří se procesu synchronizace zúčastní. Jde o statické třídy *Synchronization*, *Mapping*, *AndroidDb* a *ServerUtilities* a o objekty *HelperSql a ServerInstance*.

Fáze synchronizace jsou následující:

[6] Dostupný na adrese: www.openmobilealliance.org/syncml

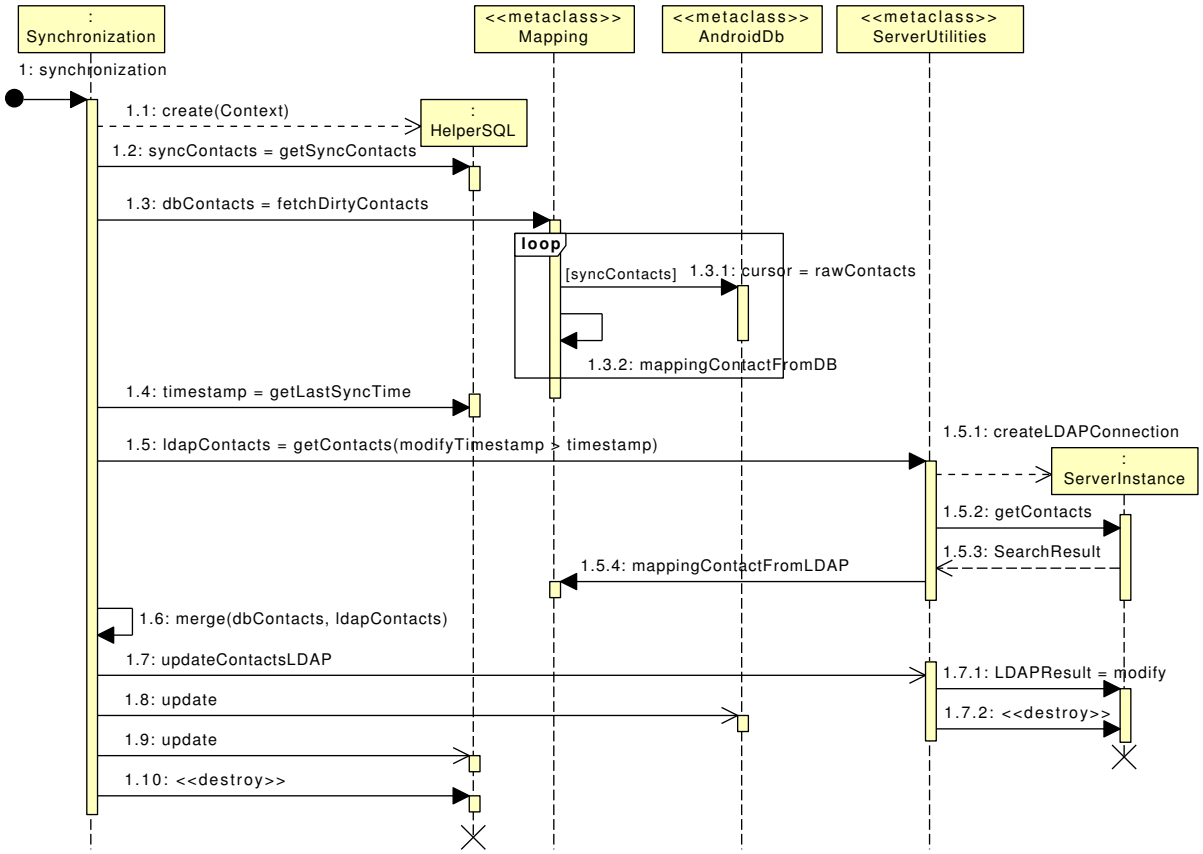
```
1  C^all_server = server.getContacts(timestamp)
2  C^all_local = local.getContactsModified(timestamp)
3  C_conflict = C^all_server ∩ C^all_local
4  C_server = C^all_server − C_conflict
5  C_local = C^all_local − C_conflict
6  C^conflict_server = ∅
7  C^conflict_loacl = ∅
8  for (contact : C_conflict) {
9     contact_local = C^all_local.get(contact.uuid)
10    contact_server = C^all_server.get(contact.uuid)
11    if (contact_local ≠ contact_server) {
12       if (contact_local.modifyTime > contact_server.modifyTime) {
13          C^conflict_local.add(contact_local)
14       } else {
15          C^conflict_server.add(contact_server)
16       }
17    } else {
18       C_local.add(contact)
19       C_server.add(contact)
20    }
21 }
22 C_local = C_local ∪ C^conflict_server
23 C_server = C_server ∪ C^conflict_local
24 server.uploadContacts(C_local)
25 local.updateContacts(C_server)
26 local.markCleanContacts(C_local)
27 createTimestamp()
```

Algoritmus 4.1: Sycnhonizace. Kostra převzata z [15].

1. Vstupní zpráva, kterou je spuštěna samotná synchronizace.
   1.1. Vytvořena instance HelperSql.
   1.2. Získán seznam kontaktů.
   1.3. Kontakty, které se změnily. Nad každým z nich jsou provedeny následující operace:
      1.3.1. Z databáze *kontakt provideru* jsou získány všechny data o daném kontaktu.
      1.3.2. Data jsou namapovány do objektu GoogleContact.
   1.4. Z databáze je získáno časové razítko poslední úspěšné synchronizace.
   1.5. Ze serveru LDAP jsou získány kontakty, které splňují následující podmínku:

$$modifyTimestamp > timestamp,$$

   neboli jsou vrácena pouze ta data, která byla modifikována od poslední synchro-



Obrázek 4.6: Synchronizační proces.

nizace:

1.5.1. Vytvořena instance LDAP připojení.

1.5.2. Vyhledávací dotaz na server.

1.5.3. Získán *SearchResult*.

1.5.4. Přemapování ze SearchResult do seznamu GoogleContacts.

1.6. Spojení kontaktů z databáze a ze serveru.

1.7. Aktualizace dat na serveru

1.7.1. Na server je zaslán příkaz *modify* s aktualizovanými daty nebo novými kontakty.

1.7.2. Zrušení instance připojení LDAP.

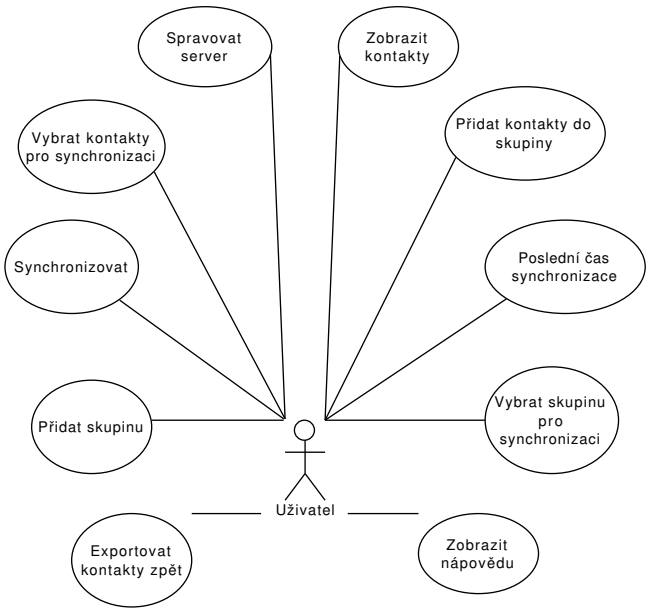
1.8. Aktualizace databáze s novými údaji.Zrušení označení pro změněné kontakty ($dirty = 0$). Natrvalo odstranění kontaktů z databáze (pokud $deleted == 0$).

1.9. Nastavení nového časového razítka.

1.10. Zrušení instance připojení k databázi.

4.5 Případy použití

V této části bude prezentován případ užití, který názorně ukazuje možnosti aplikace. Ze specifikace požadavků z kapitoly *Požadavky a analýza* 3 vyplynula pouze jedna role (uživatel). Z obrázku 4.7 je patrné, že uživatel bude převážně pracovat s kontakty nebo se serverem.



Obrázek 4.7: Případ užití.

K základní a první operaci, kterou bude nutné při použití této aplikace vykonat je nastavení synchronizačního serveru. Pro tuto akci je nutné znát, jak základní přihlašovací údaje, tak především adresu na které je LDAP server provozován.

Po přidání a nastavení serveru nastává situace, ve které by uživatel měl vybrat kontakty, které se budou synchronizovat. Na výběr má ze dvou možností. Buď si kontakty, pro synchronizaci vybere prostřednictvím skupin a jejich kontaktů nebo přímo ze seznamu kontaktů. Po této akci má již uživatel vše nastaveno a o synchronizaci se nemusí starat. Ta je spuštěna automaticky *Sync adaptérem* komponentou z Androidu.

Při odebrání aplikace nebo v situaci, kdy uživatel již nebude chtít využívat synchronizační služby, by se kontakty měly vrátit do původního stavu. To znamená, že by se měly převést na účet, na kterém byly před použitím aplikace. K tomuto účelu slouží údaje z tabulky *data*, která již byla popsaná v části zabývající se databází.

Případ užití: „Spravovat server“

Následující část se zaměřuje na případy užití „Spravovat server“. Bude zde nastíněn proces přidání nového serveru a některé výjimky, které při této činnosti mohou nastat.

Obrazovky, které umožňují přidat nový server, jsou zobrazeny pouze v těchto situacích:

1. Při prvním spuštění aplikace bude uživatel proveden základním nastavením. V jeho průběhu bude nutné zadat údaje pro vytvoření spojení se serverem.

2. Při odstranění serveru a použití nabídky server, bude uživatel proveden stejným procesem jako při prvním spuštění.

Na možnost spravovat server se uživatel dostane přímo z hlavní nabídky aplikace. Tato položka tvoří nezbytnou součást, jejíž správné fungování je nutné k běhu aplikace. V tabulce 4.1 je uvedena specifikace případu užití, kdy se uživatel pokouší přidat nový server (první start aplikace).

K základním předpokladům patří znalost přístupových údajů pro vytvoření nového spojení se serverem. Po zadání adresy serveru, takzvaného *base DN* a čísla portu, přes který se bude komunikovat, je možné otestovat stav připojení. V případě využití zabezpečené komunikace, ať už SSL/TLS nebo StartTLS je nutné zadat i přihlašovací jméno a heslo. Bez těchto údajů nebude možné server přidat ani otestovat spojení s LDAP. V situaci, kdy uživatel tuto aplikace ještě neměl nikdy nainstalovanou a využívá zabezpečený přenos, je nutné přijmout veřejný serverový certifikát. To je za předpokladu, že server nevyužívá certifikáty podepsané jednou z důvěryhodných autorit. Pokud certifikát nebude označen jako důvěryhodný nebude možné dále pokračovat. V kladném případě, je uložen v adresářovém podprostoru aplikace a uživatel již nebude ověřováním opět obtěžován.

V případu užití jsou možné následující výjimky:

1. Selhání spojení.
2. Nepřijetí certifikátu.
3. Storno.

Výše uvedený výčet výjimek je pouze ilustrativní, nesnaží se pokrýt všechny možné situace, jež by mohly nastat. V následujících částech popsány jednotlivé výjimky, jak prostřednictvím tabulek, tak i vysvětlujícím textem.

ID:	1
Název:	**Spravovat server**
Popis:	První spuštění aplikace, přidání serveru.
Primární aktéři:	Uživatel
Sekundární aktéři:	Žádní
Předpoklady:	Uživatel má a zná přístupové údaje k serveru.
Následné podmínky:	1. Jsou zobrazeny informace o daném serveru, které jsou k dispozici pro úpravu. 2. Je připravena možnost pro zadání údajů nového serveru.
Akce pro spuštění:	Uživatel spustí aplikace.
Hlavní tok:	1. Aplikace zobrazí uvítací obrazovku. 2. Uživatel vybere „Přidat server“. (a) Uživatel vybere zabezpečené připojení prostřednictvím StartTLS. (b) Uživatel vyplní potřebné údaje.
Alternativní toky:	Žádné
Výjimky:	1. Selhání spojení 2. Nepřijetí certifikátu 3. Storno
Frekvence:	Střední
Speciální požadavky:	Žádné

Tabulka 4.1: Případ užití: Spravovat server.

Výjimka případu užití: „Spravovat server - Selhání spojení“

Při zadání špatných údajů nebo nefunkčnosti internetového připojení je možné vyhodit různé výjimky. Vyvolání výjimky nazvané *Selhání spojení* je zobrazeno v tabulce 4.2. Ke vzniku této situace dojde v době, kdy aplikace nedokáže navázat spojení se vzdáleným LDAP serverem. Po neúspěšné snaze o komunikaci se serverem bude uživatel o nastalé situaci informován a přesunut zpět na obrazovku pro zadání přihlašovacích údajů.

Výjimka případu užití: „Spravovat server - Nepřijetí certifikátu“

Pro zabezpečenou komunikaci mezi serverem a klientem je využíváno SSL/TLS nebo StartTLS. U obou těchto typů je nutné pracovat s certifikáty serveru. Za podmínek, kdy server využívá nepodepsaný certifikát, je uživatel vyzván k jeho přijetí. Při zamítnutí je vyvolána výjimka popsaná v tabulce 4.3.

ID:	1.E.1
Název:	**Spravovat server: Selhání spojení**
Popis:	Aplikace se nedokáže spojit se vzdáleným serverem.
Primární aktéři:	Uživatel, server
Sekundární aktéři:	Žádní
Předpoklady:	1. Po zadání údajů se nepovede navázat spojení se serverem.
Následné podmínky:	1. (a) Jsou zobrazeny informace o daném serveru, které jsou k dispozici pro úpravu. (b) Je připravena možnost pro zadání údajů nového serveru.
Akce pro spuštění:	Selhání v kroku „Přidat server“ případu užití 4.1.
Hlavní tok:	1. Aplikace informuje uživatele o selhání navázání spojení se serverem. 2. Systém vrátí uživatele do místa, pro kontrolu údajů.
Alternativní toky:	Žádné
Frekvence:	Zřídka

Tabulka 4.2: Výjimka případu užití: Spravovat server – Selhání spojení.

ID:	1.E.2
Název:	**Spravovat server: Nepřijetí certifikátu**
Popis:	Uživatel zamítne nepodepsaný certifikát.
Primární aktéři:	Uživatel
Sekundární aktéři:	Žádní
Předpoklady:	Uživatel po zadání přihlašovacích údajů zvolí „Přidat server“.
Následné podmínky:	Správa serveru, server nebyl přidán.
Akce pro spuštění:	Uživatel zamítne nepodepsaný certifikát.
Hlavní tok:	Systém vrátí uživatele do místa, pro kontrolu údajů.
Alternativní toky:	Žádné
Frekvence:	Zřídka

Tabulka 4.3: Výjimka případu užití: Spravovat server – Zamítnutí certifikátu.

Výjimka případu užití: „Spravovat server - Storno“

Tabulka 4.4 obsahuje výjimku „Storno“. Ukazuje případ, kdy uživatel při přidávání severu akci ukončí. Po stornování prováděné akce je aplikace ukončena a nachází se ve stavu, stejném jako před započetím případu užití. Výsledkem je, že server nebyl přidán a při

dalším spuštění aplikace je nezbytné nastavení provést opětovně.

ID:	1.E.2
Název:	**Spravovat server: Storno**
Popis:	Uživatel ukončí případ užití 4.1.
Primární aktéři:	Uživatel
Sekundární aktéři:	Žádní
Předpoklady:	Žádné
Následné podmínky:	Správa serveru, server nebyl přidán.
Akce pro spuštění:	Uživatel zvolí „zpět“ kdykoli v průběhu hlavního toku případu 4.1.
Hlavní tok:	Aplikace vrátí uživatele do místa, odkud vyvolal případ užití (ukončí aplikaci) 4.1.
Alternativní toky:	Žádné
Frekvence:	Zřídka

Tabulka 4.4: Výjimka případu užití: Spravovat server – Storno.

4.6 Obrazovky

Po návrhu tzv. „backendu“ následuje navržení grafické uživatelské rozhraní. Tato část je převážně tvořena obrázky vytvořenými v aplikaci *WireFrameSketcher*[7], která je pro návrhy obrazovek určena.

Navigační panel

Každá obrazovka obsahuje navigační panel, který prostřednictvím ikon zpřístupňuje funkce jako jsou nápověda a nastavení. Obrázek 4.8 zobrazuje návrh dvou různých navigačních panelů. První, položen výše, je vyříznut z hlavního menu. Zleva obsahuje ikonu a název aplikace. Dále to jsou ikony nápověda a nastavení, které jsou dostupné z každé obrazovky.

Obrázek 4.8: Navigační panel aplikací.

Druhý navigační panel pochází z obrazovky, na které je možné vybrat kontakty pro synchronizaci. Na levé straně se nachází šipka zpět, prostřednictvím ní je možné se vrátit na její rodičovskou obrazovku. Text vedle ikony aplikace je určen pro určení pozice, kde se uživatel v aplikaci nachází. Stisknutím ikony, složené ze dvou šipek do tvaru kolečka,

[7] dostupná na http://wireframesketcher.com/

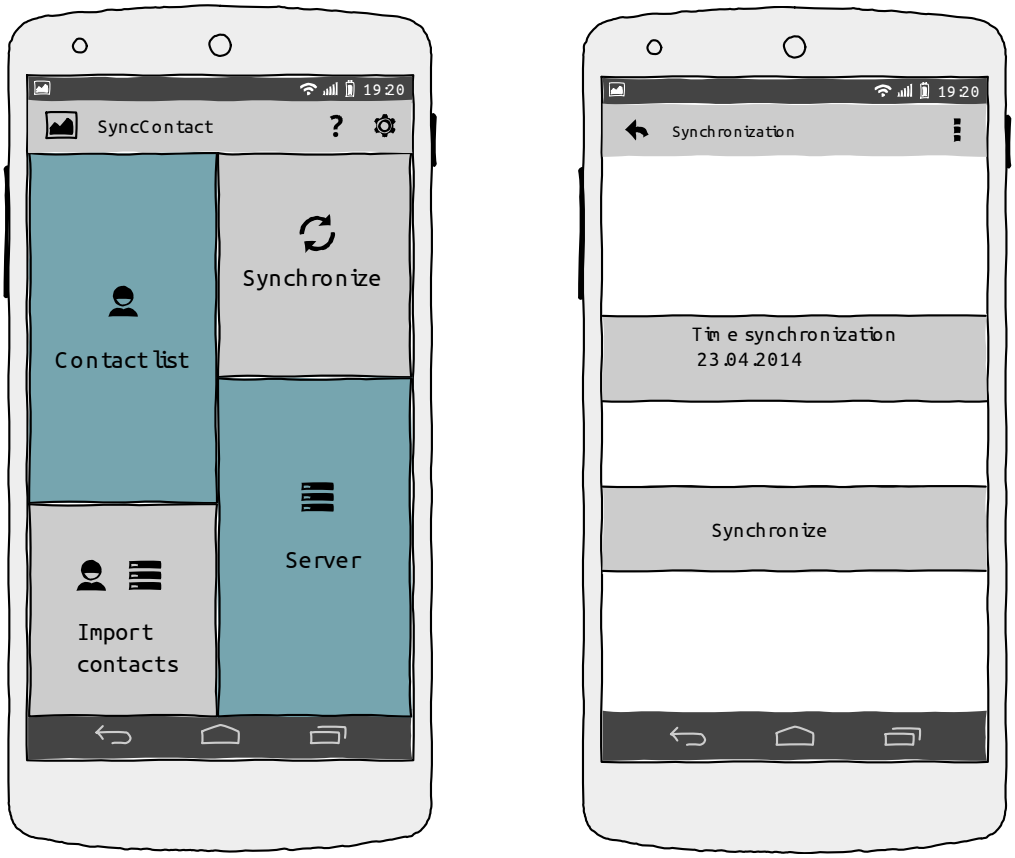
se vyvolá funkce, která zaktualizuje právě zobrazený seznam kontaktů a záznamů. Tato operace je časově velice náročná, jelikož se pracuje, jak s databází z kontakt provideru, tak i databází podpůrnou. Poslední ikona (osoby se znakem plus) je nadefinována pro přidávání nové skupiny a k výběru kontaktů, které do ní budou patřit.

V případě, kdy je obrazovka mobilního zařízení malá, jsou ikony schovány do menu. To je možné vyvolat tlačítkem umístěným napravo nebo použití hardwarového tlačítka pro menu.

Obrazovky

Okna všech obrazovek jsou rozdělena na dvě oblasti, přibližně v poměru 1:10. Menší část, umístěná při horním okraji displeje, byla popsána výše, je určena pro název aplikace a pro ikony zpřístupňující různé funkce. Druhá část slouží pro aktivní data.

Jako první je ukázán návrh úvodní obrazovky a synchronizace 4.9. Hlavní aktivita 4.9a slouží jako vstupní brána pro ostatní funkce. Je tvořena čtyřmi barevně odlišenými dlaždicemi. Obrázek 4.9b ukazuje informace o poslední synchronizaci a nabízí možnost provést synchronizaci ručně.



(a) Úvodní obrazovka (b) Synchronizace

Obrázek 4.9: Obrazovky 1.

Další dvě obrazovky jsou v příloze E na obrázku E.1. Jedná se o návrh kontaktů E.1a a zobrazení serveru E.1b.

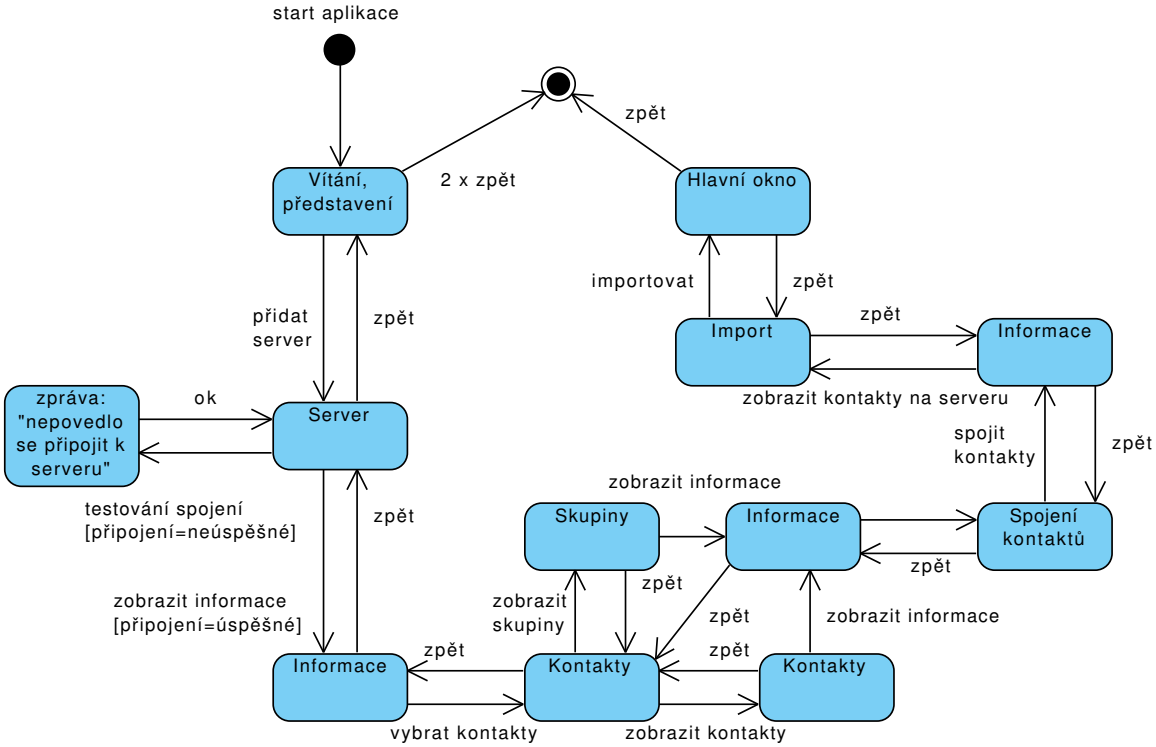
4.7 Diagram návaznosti obrazovek

V předešlé části bylo navrženo grafické uživatelské rozhraní pro aplikaci bežící na operačním systému Android 4.0 a vyšší. Byly zde uvedeny pouze obrazovky, nyní k nim bude přidána jejich návaznost. Bude ukázáno v jakém vztahu jsou jednotlivé *aktivity*[8].

Na dvou schématech 4.10 a 4.11 jsou předvedeny všechny možné obrazovky a vztahy mezi nimi.

První spuštění

Při prvním spuštění aplikace je uživatel proveden procesem, při kterém je vytvořeno připojení k serveru a jsou vybrány kontakty, jež se budou synchronizovat. Těmto krokům odpovídá schéma 4.10. V průběhu tohoto průvodce bude uživatel vyzván ke spojení kontaktů, které jsou na mobilním zařízení s těmi, které jsou na serveru. V případě, kdy uživatel podobné kontakty neoznačí jako stejné, budou považovány za odlišné a bude s nimi také tak zacházeno. Na poslední obrazovce bude možné vybrat si kontakty, jež se mají do telefonu naimportovat.

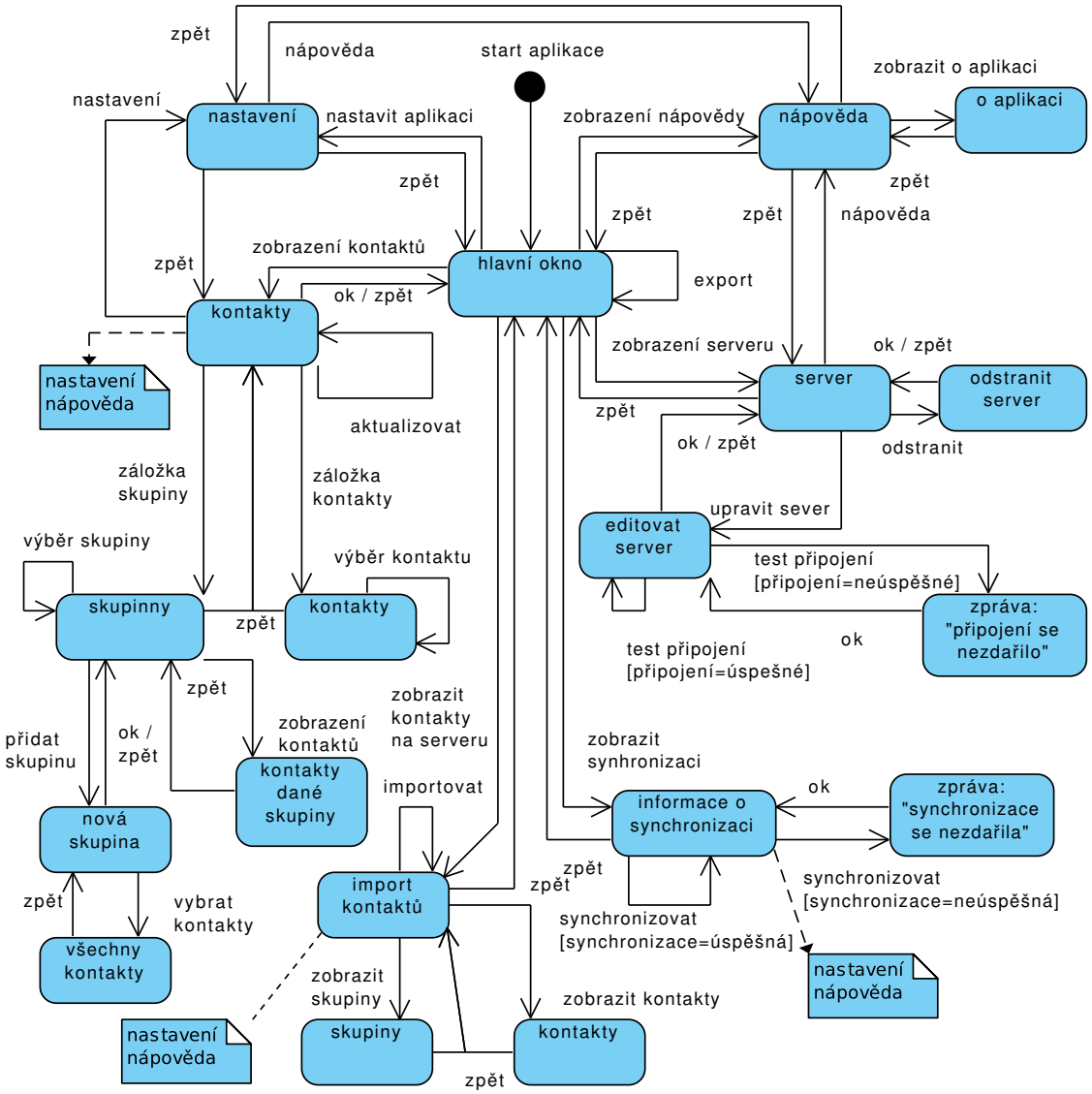


Obrázek 4.10: Diagram návaznosti obrazovek – první spuštění.

[8] Aktivitu neboli „activity“ lze v prostřední Android chápat jako obrazovku. Jde o prostředek prostřednictvím něhož lze komunikovat s uživatelem.

Další spuštění

Na diagramu 4.11 jsou již zobrazeny všechny obrazovky (vyjma prvního spuštění) a kroky, které je nutné provést při přechodu mezi nimi. Z *hlavního okna* je přístupná nabídka *ex-*



Obrázek 4.11: Diagram návaznosti obrazovek.

portovat, prostřednictvím ní je možné převést kontakty zpět na původní účet.

Kapitola 5

Implementace

Následující kapitola je zaměřena na vývoj aplikace. Klade si za cíl popsat základní informace o použité technologii a o omezeních, která byla provedena či o dalších zásadních rozhodnutí. První část kapitoly je věnována operačnímu systému, pro který je aplikace navržena (Android) [18, 14, 6, 11]. Jsou zde uvedeny některé využité podpůrné nástroje či popsána kostra aplikace.

Ve druhé části jsou rozebrány implementační detaily, které se přímo týkají serveru. Jde například o tvorbu schématu [4, 7] či o práci s certifikáty.

5.1 Aplikační část

Aplikace je postavena na Android SDK. V práci je využívána nejnovější[1] dostupná verze označena *Android 4.4 (API 19)*. Vývoj probíhal na operačním systému Ubuntu 13.10 64-bit (java verze 1.7.0) v aplikaci Android Developer Tools[2], která je postavena na software Eclipse verze 22.3.

K dalším podpůrným nástrojům lze zařadit SQLite Debugger[3], který byl nainstalován na mobilním zařízení, jež sloužilo jako testovací prostředí. Jelikož se v práci pracuje s databází kontaktů, která je vlastněna aplikací *contacts provider* a ta je přímo zabudována do systému Android, byl nutný zásah do samotného systému. Šlo o získání administrátorských práv pro určité aplikace. Tohoto kroku bylo docíleno nainstalováním tzv. *CyanogenMod*, na němž je již „root“ proveden a je k dispozici *Superuser*. Prostřednictvím oprávnění administrátora lze získat přístup do systémových částí, kde je možné na data jak nahlížet, tak je i upravovat. Tímto aplikace SQLite Debugger mohla získat potřebné oprávnění pro přístup k databázi *contacts2.db*.

Verze

Aplikace je určena pro Android od verze 4.0 a vyšší. Tento krok byl zvolen na základě rozšířenosti používání verzí operačního systému Android. Podklady pro toto rozhodnutí byly čerpány ze stránek, jež se věnují vývoji platformy [3].

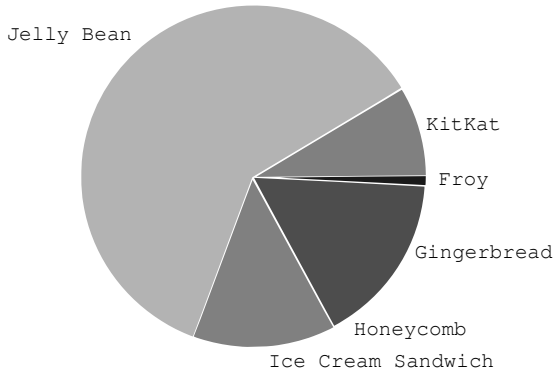
Jak je zřejmé z obrázku, 5.1 verze Androidu nižší jak *Ice Cream Sandwich*, což je označení pro API 15, jsou zastoupeny pouze 17 %. Převážná část koláčového diagramu

[1] V době psaní této práce dne 27. května 2014.

[2] Dostupný na adrese: http://developer.android.com/sdk/index.html

[3] Dostupný na adrese: https://play.google.com/store/apps/details?id=oliver.ehrenmueller.dbadmin&hl=cs

je tvořena *Jelly Bean*, což je kódové jméno pro verze 4.1.x až 4.3. Nejnovější *KitKat* je obsažen bezmála na 10 % zařízení využívající systém Android. Sečtou-li se všechny verze 4.0 +, dostane se procentuální zastoupení něco málo přes osmdesát procent.



Obrázek 5.1: Zastoupení verzí Androidu v mobilních zařízeních [3].

5.1.1 Knihovny a Framework

Při implementaci aplikace byla využívána knihovna *UnboundID*[4] a framework *Androidannotations*[5].

Pro komunikaci se serverem v prostředí Java existuje řada knihoven, které jsou pro tuto činnost vytvořeny. Jako příklad lze uvést *JNDI* nebo *Netscape Directory SDK for Java*. V porovnání s ostatními knihovnami UnboundID působil propracovanějším dojmem. Ať už se jednalo o nespočet příkladů, dokumentaci či projektovou dokumentaci.

Velkou roli při výběru této knihovny bylo zjištění, že většina aplikací pro Android, které využívají komunikaci s LDAP serverem, je postaveno právě na této knihovně. Existuje i volně šiřitelný projekt (pod licencí Apache) LDAP-Sync, o kterém již bylo psáno v části zabývající se analýzou. Na domovských stránkách[6] lze nalézt i soubory se zdrojovým kódem.

Android aplikace jsou vyvíjeny v „osekaném“ jazyce Java. V souvislosti s vývojem v tomto objektovém jazyce jsou často využívány různé frameworky. Taktéž Androidannotations [1] je freamwork, jež si klade za cíl zpříjemnit vývoj. Snaží se toho dosáhnout zjednodušením a zpřehledněním kódu prostřednictvím využívání anotací. Jako příklad lze uvést anotace pro rozšíření komponent: *@EActivity, @EFragment, ...*. V hojné míře jsou v práci využívány anotace zaměřené na vlákna a to především *@Background*.

Dle zadání práce je nutné, aby aplikace byla zpřístupněna pod licencí GPL. Využití knihovny UnboundID ani frameworku Androidannotations s touto podmínkou nekolidují. Obě rozšíření lze v open sourcové implementaci využít.

5.1.2 Architektura

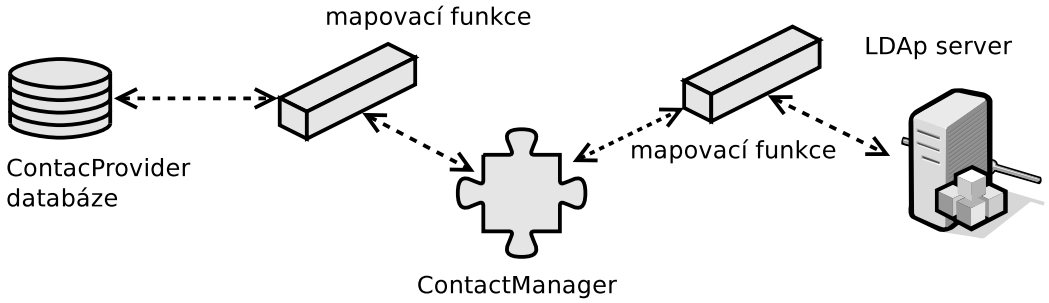
Při implementaci aplikace byly využity základní prvky objektově-orientovaného programování. Každá třída zde zastupuje jednoznačně oddělitelné elementy. Ať už se jedná o bloky

[4] Dostupná ze stránek: https://www.unboundid.com/products/ldapsdk/
[5] Dostupný ze stránek: https://github.com/excilys/androidannotations
[6] https://github.com/weisserd/LDAP-Sync

komunikující s databází nebo o části, které se starají o výměnu informací mezi zařízením a LDAP serverem.

Komunikace s databází se nachází v samostatné třídě, která zajišťuje provázanost mezi ní a daty. Pomocí funkcí jsou data, získaná ze synchronizačního procesu, přenášena a ukládána do jednotlivých tabulek databáze. Při zahájení činnosti aplikace je zajištěno navázání připojení k databázi a provedení nutných inicializačních kroků. Jedná se o přesunutí kontaktů do nového účtu a tvorbu záznamů v tabulce *data* pro uložení metadat o předchozím účtu nebo o kontrolu existence těchto entit při následném spuštění. Taktéž jsou nastaveny synchronizační sloupce *sync*. V případně chybějících dat jsou pro tyto sloupce údaje (uuid, timestamp) vygenerovány nově.

Základní část, kterou je možné považovat za jádro aplikace, je třída `ContactManager`. Je implementována jako jedináček (singleton) a udržuje informace o kontaktech. Neboli prostřednictvím ní je získán přístup k samotným kontaktům či skupinám. Projekt lze rozdělit do několika komponent, kde se v jeho středu nachází právě ContactManager. Schéma těchto částí je zobrazeno na obrázku 5.2. Nedílnou součástí aplikace jsou tzv. „mapovací funkce“, které mají za cíl převádět data z různých formátů mezi sebou. Jde především o části, kdy jsou data uložena v databázi v několika tabulkách nebo při situaci, kdy jsou kontakty přenášeny mezi klientem a serverem.



Obrázek 5.2: Architektura projektu.

S mapovacími funkcemi úzce spolupracují metody, které se zabývají slučováním kontaktů. Ty jsou využívány převážně při synchronizaci. Nastane-li situace, kdy je nutné kontakt aktualizovat na mobilním zařízení a již zde v určitém stavu existuje, je nutné provést sloučení.

5.1.3 Oprávnění

Při instalaci aplikace na zařízení je nutné, aby uživatel potvrdil zda souhlasí s podmínkami. V této části je mu zobrazena i velice důležitá část, která se týká oprávnění pro danou aplikaci. Je mu vypsán seznam oprávnění, které instalovaná aplikace vyžaduje pro svoji činnost. V tomto ohledu je vyvíjená aplikace velice náročná. Pro svůj správný běh vyžaduje mnoho oprávnění. Řada z nich je zobrazena níže a je popsán důvod jejich výskytu. Především jde o možnost přistupovat ke kontaktům, k účtům a k synchronizačním údajům.

GET_ACCOUNTS a MANAGE_ACCOUNTS - dává aplikaci přístup ke všem účtům na telefonu. Povolení je vyžadováno za účelem vytvoření nového účtu a k získání zpět informací o nově přidaném účtu.

READ_CONTACTS a WRITE_CONTACTS - povoluje aplikaci k manipulaci s kontakty na telefonu. Oprávnění je vyžadováno za účelem synchronizace kontaktů. To představuje procesy jako je jejich editace, tvorba nového či mazání.

5.1.4 Export kontaktů

Jak již bylo nastíněno výše, v kapitole návrhu popisující strukturu databáze, pro korektní fungování synchronizace je nutné převést kontakty do nově vytvořeného účtu. Tento požadavek je postaven na faktu nutnosti správného fungování položky *dirty* v tabulce *raw_contacts*. Tento údaj nese informace o tom zdali byl daný záznam změněn či ne. V případě, že je kontakt pod účtem této aplikace, lze předpokládat, že hodnotu *dirty* vynuluje pouze její synchronizační adaptér.

5.2 Serverová část

V následující části jsou popsány kroky, jež je nutné provést pro správné nastavení LDAP serveru. Další podrobnosti jako je instalace serveru a přidání nového schématu lze nalézt v příloze .

Identifikace objektů

Pro jednoznačnou definici objektů a atributů, které společně tvoří schémata, jsou využívány tzv. *Object Identifier Definitions* (OID). OID neboli identifikace objektů rozšiřuje definici jazyka pro tvorbu LDAP schémat. Jeho přínosem není přidávat novou funkcionalitu, pouze se snaží usnadnit orientaci při práci se schématem. Každý atribut, objekt nebo i pravidla definující shodu musí obsahovat OID, které je tvořeno sekvencí celých čísel oddělených mezi sebou tečkami (.). Kombinace a pořadí těchto číslic není náhodné, jejich struktura by měla reprezentovat rodokmen objektu. OID je tvořeno z následujících částí:

- základní OID,
- číslo typu,
- číslo položky.

Při tvorbě nového objektu, u něhož je předpoklad, že bude využíván v kombinaci s jinými objekty je doporučeno nechat si vygenerovat soukromou základní část OID. Přidělování základních OID jsou přenechány tzv. jmenným autoritám (například IANA[7]). Pro schéma *SyncContact* vytvořené pro potřeby této práce bylo přiřazeno číslo `43624`. Jak je patrné ze schématu 5.3, číslo přiřazené autoritou IANA a kombinace čísel 1.3.6.1.4.1.?, kde ? je nahrazen přiděleným číslem tvoří základní část. Objekty obsahující tuto základní část jsou jedinečné v globálním měřítku.

```
1.3.6.1.4.1.? + 43624 => 1.3.6.1.4.1.43624.x.y
```

Obrázek 5.3: Schéma tvorby OID.

Pro zbylé dvě části (položka x a typ y) je doporučeno používat následující syntaxi [7]:

[7] Více na stránkách: http://www.iana.org/assignments/enterprise-numbers

- LDAP syntaxe (1),
- pravidla na shodu (2),
- typy atributů (3),
- třídy objektů (4),
- podporované utility (5),
- mechanismy protokolu (9),
- kontroléry (10),
- rozšiřující operace (11).

5.2.1 Schéma a DIT

Pro LDAP server bylo vytvořeno nové schéma, které opisuje všechny doposud nabízené atributy pro kontakty v Android SDK. Počet atributů se pohybuje přes 100, a i když by bylo možné některé z nich využít již z definovaných tříd, pro přehlednost byly vytvořeny znovu, samozřejmě s jiným OID. Většina těchto atributů dědí z atributu *name* (2.5.4.41) definovaném v RFC 2252. Základní a také jediná třída *googleContact* dědí ze třídy *top*. Zastřešuje všechny atributy jako nepovinné, pouze *UUID* atribut je povinný z důvodu využití pro RDN.

Atribut UUID[8](Univerzální Jedinečný IDentifikátor) slouží pro umístění a rozlišení kontaktů a skupin na serveru. Programové generování je přenecháno na implementaci v Javě. Jde o třídu umístěnou v balíčku *java.util*, který využívá UUID standardu definovaném v RFC 4122[9] varianty druhé.

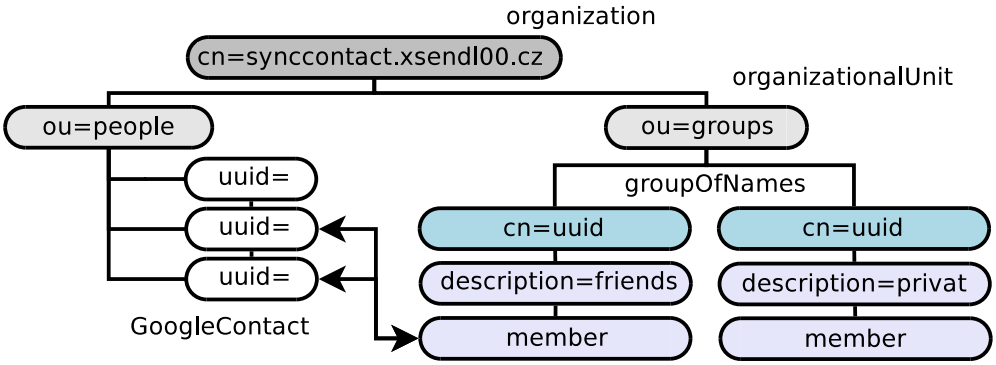
Schéma je umístěno na přiloženém CD v adresáři `./openLDAP/ldif/`. Definice třídy bez atributů je možné nalézt v příloze C.

Pro ukládání záznamů typu *skupina* je využíváno schéma *groupOfNames*. Obsahuje pouze tři atributy. První je povinný, tzv. *cn* (commonName), který slouží jako RDN a v tomto případě obsahuje hodnoty UUID. Druhý parametr *description* (popis) je využit pro jméno tabulky převzaté z mobilního zařízení. Poslední položka s názvem *member* (člen) je určena pro odkaz na kontakty umístěné v dané skupině. Za tímto účelem je výskyt tohoto parametru vícenásobný.

Po popisu struktury pro uchování záznamů typu kontakt a skupina je nyní možné prezentovat celou stromovou strukturu. K tomuto účelu slouží obrázek 5.4. Základní část neboli kořen stromu (cn=synccontact.xsendl00.cz) je typu *organization*. Tuto položku bude nutné zadat při vytváření spojení v mobilní aplikaci. Ostatní struktura je již pevně daná a není možné použít jinou.

Položky v organizaci, které jsou ve stromové struktuře ihned pod kořenem, (*organizationalUnit*) jsou dvě (people a groups). Záznam lidé slouží jako adresář pro všechny kontakty, jež jsou plně kvalifikované pomocí uuid a DN lidé a jsou typu GoogleContact. Druhá položka, skupiny, obsahuje jednotlivé skupiny třídy *groupsOfNames*. Jsou kvalifikované taktéž pomocí uuid a nadřazeného DN skupin. Skupina obsahuje vícehodnotový parametr *member*, který odkazuje na kontakty patřící do dané skupiny. Na obrázku je tato skutečnost znázorněna pomocí oboustranných šipek.

[8]UUID - Universally Unique IDentifier
[9]Dostupné na adrese: http://www.ietf.org/rfc/rfc4122.txt



Obrázek 5.4: Stromová architektura LDAP.

Postup jak nově vytvořené schéma přidat do již existující instalace OpenLDAP je popsán v příloze . Návod je určen pro OpenLDAP verze 2.4.31 a pro operační systém Ubuntu 13.10.

5.2.2 Seznam řízeného přístupu

Syntaxe seznamu řízeného přístupu byla podrobně popsána v části věnované bezpečnostnímu modelu 2.4. Následující text si klade za cíl prezentovat vybraná pravidla, jež jsou v práci používána a zdůvodnit jejich existenci.

5.3 Omezení aplikace

V mobilním prostředí existují tři druhy kontaktů. Kontakty uložené na SIM kartě, v mobilním zařízení a na účtech jako je google.com nebo skype. Po dohodě s vedoucím diplomové práce jsou do práce zahrnuty pouze plnohodnotné kontakty, mezi něž kontakty na SIM kartě nelze řadit.

Z výše uvedeného předpokladu, je zřejmé, že kontakty na mobilním telefonu jsou uloženy pod určitým účtem. Při nainstalování aplikace *SyncContact* je vytvořen nový účet *syncContact.cz*. Aby bylo možné využívat prostředky nabízené *providerem kontaktů* je nutné stávající kontakty přidat pod nově vytvořený účet. Tento proces je proveden automaticky při prvním spuštění aplikace a po průchodu základním nastavením. V případě, že by tento krok nebyl proveden, bylo by nutné všechny kontakty v mobilním zařízení duplikovat, což po domluvě s vedoucím práce bylo označeno za nevhodný postup.

Před odstraněním aplikace je nutné kontakty `převést zpět do původního účtu`. Toho je možné dosáhnout prostřednictvím nabídky na úvodní obrazovce, vyvolané stiskem tlačítka menu. Tímto krokem budou kontakty převedeny do stavu před použitím aplikace. V případě, že bude aplikace odinstalována bez tohoto kroku, budou *všechny kontakty odstraněny s ní*. Toto chování je u Androidu zcela běžné. Ten při odinstalování aplikace odebere i všechny přidružené účty, kontakty či data uložená jak v primárním prostoru tak v prostoru na paměťové kartě.

Při odstranění připojení nebo účtu ať už přímo z aplikace *SyncContact* nebo přes systémovou položku *účty a synchronizace* jsou kontakty převedeny do původního stavu automaticky.

Synchronizované položky

Operační sytém Android pro atributy u kontaktů využívá tabulku *data*. Jednotlivé položky jsou mezi sebou rozlišeny prostřednictvím tzv. *mimeType*, neboli typu dat. Tyto hodnoty jsou v Android SDK pevně dané a jsou podporovány vyvíjenou aplikací. Konkrétně se jedná o: email, událost, adresy komunikačních kanálů, přezdívky, organizace, telefonní čísla, vztah osoby k jiným osobám, adresu, osobní data a webové adresy.

Množinu typů atributů u kontaktů lze rozšířit o vlastní nové druhy. Tuto skutečnost například využívá aplikace Skype, která do tabulky *mimetypes* přidává vlastní položky. Je zřejmé, že bez programové úpravy aplikace nelze rozšířit synchronizaci o položky, jež nejsou pevně zabudovány v Android SDK.

Kapitola 6

Diskuze

Následující části jsou zaměřeny na hodnocení výsledků a na možnosti dalšího rozšíření práce.

6.1 Zhodnocení

Z první analýzy se se zdálo, že využití LDAP serveru jako cloudu pro zálohu kontaktů, je ideální nápad. V průběhu práce a při zjištění možností mobilního zařízení již tento výběr za ideální určit nelze. Za největší nevýhody lze považovat fakt, že server je tzv. „statický“. Nelze tedy využít jeho prostředků jiným způsobem než pro uložení, případně pro výběr dat. Práci spojenou se synchronizací je nutné vykonávat na mobilním zařízení, což se ukázalo jako ne zrovna jednoduché.

Problémy mohou nastat převážně v situaci, kdy je nutné řešit konflikty, neboli když jsou data modifikována, jak na serveru tak i v mobilním zařízení. Tato situace je řešena upřednostňováním dat ze serveru před kontakty u klienta. Od verze API 18 je k dispozici časový údaj, kdy byl kontakt na mobilním zařízení změněn. Tento údaje by bylo možné použít pro přesnější rozhodování, která data jsou novější a zachovat právě je. Jelikož, je ale podporována již verze 4.0 není tento atribut u starších verzí využit.

Jako řešení výše uvedeného problému by bylo možné místo LDAP serveru využít specializovaný server, který by dokázal řešit konflikty sám. Nabízí se možnost použití standardu *SyncML* (více informací v části zabývající se synchronizací) spolu s aplikačním serverem, například *Apache Tomcat*. U tohoto řešení by bylo nutné na implementovat serverovou část, která by se starala o případné konflikty, „mergování“ i ukládání dat na straně serveru. K tomuto účelu se nabízí využít některou „odlehčenou“ databází například *SQLite* nebo jiné.

Testování

Vzájemná spolupráce mobilní aplikace a serveru byla testována přibližně na 150 kontaktech. Každý kontakt obsahoval jméno, telefonní číslo a poštovní adresu. Některé byly rozšířené o adresu bydliště či zaměstnání. Pouze malé procento kontaktů obsahovalo více jak 20 záznamů.

Pro testování byl použit domácí stolní počítač na němž běžela adresářová služba. Mobilní zařízení byly zastoupeny těmito modely: ZTE Blade (verze Androidu 4.2.2), tablet Nexus 7 2013 (4.4), Sony Ericsson WT19i (4.1) a několik virtuálních zařízení. U fyzických

telefonů testování probíhalo jak v lokální síti tak i s mobilním připojením. Virtuální zařízení sloužila převážně k otestování stability aplikace.

Samotná výkonnost aplikace je složitě měřitelná. Lze ale říci, že čas potřebný k synchronizaci je přímo-úměrný počtu kontaktů a jejich velikosti. Z výsledků testů v tabulce 6.1, které byly provedeny při vývoji lze vyvodit, že nejdelší čas v celé synchronizaci zaujímá práce s databází. A to i přes to, že při vývoji byl kladen důraz na co nejmenší počet přístupů do databáze. Je zřejmé, že tuto část lze považovat za slabé místo a je vhodné k optimalizaci.

Čas naměřený při importu kontaktů tvořil 90 % celkového času potřebného pro celý proces. Rozdíly v tabulce nejsou velké 6.1, nicméně je nutné uvést, že synchronizace s aplikací SyncContact probíhala v lokální síti s použití kanálu StartTLS. Importování dat bylo provedeno na 150 vzorcích a synchronizace na 40 změněných kontaktech. Měření bylo prováděno 10 krát a výsledky byly zprůměrovány. Pro testování časů byl použit mobilní telefon

Typ procesu	Google sync	SyncContact
import	50 s	70 s
synchronizace	20 s	30 s

Tabulka 6.1: Srovnání Google sync a SyncContact.

Zte Blade, který nelze považovat za svižné zařízení. Vlastní pouze jedno-jádrový procesor 600 MHZ a 420 MB RAM, z nichž systém (s vybranými aplikacemi v pozadí) průměrně využívá 370 MB. Velikost tabulek, se kterými se pracovalo, bylo pro *raw_contact* 50 KB pro 152 záznamů a pro *data* s 1146 záznamy 200 KB.

Jako optimalizaci souvisící s rychlostí je možné uvést pouze re-faktorizaci zdrojového kódu aplikace. Možnosti jako jsou zasílání tzv. „dump“ tabulky na server, který by data zpracoval, není v souvislosti s použitím LDAP serveru možné.

6.2 Další rozšíření

Mobilní aplikace spolu se serverem OpenLDAP vytváří systém pro synchronizaci kontaktů. Je postaven na faktu, že klientská část běží na operačním systému Android od verze 4.0. Z hlediska dalšího pokračování by se nabízela možnost upravit aplikaci takovým způsobem, aby mohla být využívána i na nižší verzi. K dosažení tohoto stavu by nebylo zapotřebí mnoho programového úsilí, ale převážně mnoho času pro odladění na danou verzi. Také bylo by nutné nalézt zařízení se starší verzí Androidu, jelikož testování aplikace v emulátoru není ideální.

Jako druhé rozšíření se nabízí možnost využití, jak vlastního adresářového schématu tak i vlastních tříd. Tato myšlenka byla ze začátku práce rozvíjena, ale při pohledu na obrazovku, jež by mapovala atributy z LDAP tříd na shodné objekty nebo parametry v prostředí Androidu, bylo od této možnosti ustoupeno.

Daleko zajímavější podnět vyšel při hledání synchronizačního algoritmu, kdy bylo zjištěno že se systém podobá verzovacím nástrojům. Jednalo by se o možnost rozšíření adresářové struktury o další adresáře, jež by reprezentovaly jednotlivé kontakty a jejich podstrom by byl tvořen verzemi daného kontaktu. Při každé synchronizaci, kdy by docházelo k nějaké změně by byl vytvořen nový záznam a ten stávající by byl přesunut do „záložní“ větve.

Tento doplněk by bylo možné řešit i jinou cestou. Záložní záznamy by mohly obsahovat pouze rozdílové atributy.

Poslední rozšíření je zaměřeno na kontakty. Bylo by vhodné implementovat aplikaci, jež by umožňovala správu kontaktů prostřednictvím webového rozhraní (backend by zůstal LDAP server). Jejími přednostmi by především mělo být přehledné a jednoduché uživatelské prostředí. Je možné namítnout, že aplikace pro správu LDAP serveru již existují, ale dle mého názoru nejsou vhodná pro tento případ. Největší nevýhodu u těchto řešení lze nalézt v jejich složitosti, nepřehlednosti a způsobu jakým jsou položky editovány.

K samotnému výběru kontaktů, které se mají synchronizovat nebylo nalezeno další rozšíření.

Kapitola 7

Závěr

V diplomové práci byla řešena problematika synchronizace kontaktů na zařízení s operačním systémem Android při využití adresářového serveru. V oblasti synchronizace existuje mnoho různých prostředků, které by se daly využít k zálohování dat. Všechny dostupné nástroje na trhu mají společnou vlastnost. Jedná se o to, že k daným datům má přístup právě subjekt, který tuto službu nabízí. Není zde stoprocentně zaručena ochrana před nežádoucími subjekty.

Cílem práce bylo poskytnout nástroj k nahrazení stávajících možností zálohování kontaktů v mobilních zařízení. Systém vytvořený v této práci je určen pro uživatele, kteří nechtějí svá data poskytovat třetí straně. K fungování celého systému je zapotřebí mít také k dispozici OpenLDAP server, jež slouží jako cloud pro uchování dat.

Jako základní funkce aplikace lze uvést možnost připojit mobilní zařízení k nakonfigurovanému serveru podporující LDAPv3 a synchronizovat s tímto adresářovým serverem předem specifikovaná data. Kontakty, které budou podléhat synchronizaci, je možné vybírat dvěma způsoby. V prvním případě jde o jednoduchý přímý výběr kontaktů ze seznamu. Druhou možností je označení celé skupiny, čímž jsou do synchronizačního procesu zahrnuty všechny kontakty, jež do dané skupiny patří. Kontakty, které jsou nově vytvořeny lokálně na mobilním zařízení a náleží do skupiny označené pro synchronizaci, jsou automaticky zahrnuty při zálohování. Tímto způsobem bude možné oddělit privátní data od korporátních a každý uživatel si bude moci určit, které kontakty budou synchronizovány.

Aplikace byla privátně navržena pro synchronizaci záznamů mezi něž se řadí atributy jako jsou: adresa, jméno email či telefonní čísla. Pro běh mobilní aplikace je nutný operační systém Android, zejména díky jeho rozšířenosti mezi uživateli. Android je vyžadován v minimální verzi 4.0. Synchronizace je zajištěna pomocí protokolu LDAP. Pro serverovou část byla použita „open sourcová“ implementace LDAP protokolu, software OpenLDAP. S využitím obou částí, mobilní aplikace a adresářového serveru, byl vytvořen systém zajišťující synchronizaci kontaktů v privátním cloudu ve vlastní režii uživatele.

Cíle práce byly úspěšně splněny. Druhá kapitola byla zaměřena na možnosti adresářového serveru. Byly zde rozebrány následující modely: informační, jmenný, funkční a bezpečnostní. Informace uvedené ve druhé části byly následně využity pro návrh mobilní aplikace a to především při výběru vhodné knihovny. Analýza požadavků na mobilní aplikaci byla obsahem třetí kapitoly. Také zde byla rozebrána stávající řešení a definovány funkční i nefunkční požadavky. V kapitole čtvrté, kterou je možné nalézt pod názvem návrh, byl představen obecný návrh aplikace s jeho popisem a návrh řešení bezpečnosti. Především zde byl kladen důraz na využití databází jak pro získání dat, tak i pro uložení dodatečných informací. Byl zde prezentován algoritmus, jež je součástí synchronizace. Zbylá část kapitoly se věnovala

diagramům a popisu případu užití, návrhu grafického uživatelského rozhraní a návaznosti obrazovek. Implementační detaily byly prezentovány v kapitole čtvrté. Šlo především o návrh stromové struktury či o výběr podpůrných nástrojů. Předmětem předposlední kapitoly bylo zhodnocení dosažených výsledků a návrh dalšího rozšíření.

Za největší problém, který byl bohužel vyřešen pouze částečně, lze považovat fakt, že při odinstalování aplikace z mobilního zařízení jsou odstraněna i všechna data s ní spojená. To se týká i všech kontaktů, které byly na začátku práce s mobilní aplikací převedeny na nově vytvořený účet. Tento postup, jak bylo zjištěno, je v systému Android běžný. Problém byl částečně zmírněn tím, že kontakty jsou automaticky převedeny ve dvou následujících případech: při odstranění účtu prostřednictvím nabídky v nastavení systému, nebo přímo v aplikaci a to konkrétně v části odebrání serveru. Při odinstalování aplikace je doporučen provést krok exportu kontaktů do předešlého účtu (pomocí nabídky menu na hlavní obrazovce aplikace).

Každý subjekt vlastnící mobilní telefon s operačním systémem Android si může vybrat sám jako volbu zálohy kontaktů zvolí. Ať už je to využití řešení od nějaké renomované společnosti, nebo možnosti nabízené přímo operačním systémem Android ve spolupráci s firmou Google, či jiné alternativy prezentované například touto prací.

Literatura

[1] Androidannotations. online, 1-9-2013, [cit. 27. května 2014].
URL https://github.com/excilys/androidannotations/wiki

[2] Contacts Provider. online, 2013, [cit. 27. května 2014].
URL http://developer.android.com/guide/topics/providers/contacts-provider.html

[3] Dashboards. online, 1-4-2014, [cit. 27. května 2014].
URL https://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

[4] LDAP for Rocket Scientists. online, 16-8-2013, [cit. 27. května 2014].
URL http://www.zytrax.com/books/ldap

[5] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1990: s. 1–84, doi:10.1109/IEEESTD.1990.101064.

[6] Allen, G.: *Android 4*. Brno: Computer Press, první vydání, 2013, 656 s., iSBN 978-80-251-3782-6.

[7] Butcher, M.: *Mastering OpenLDAP*. Birmingham: Packt, první vydání, 2007, 484 s., iSBN 978-1-847191-02-1.

[8] Carter, G.: *LDAP system administration*. Cambridge: O´Reilly, první vydání, 2003, 294 s., iSBN 1-56592-491-6.

[9] Donley, C.: *LDAP Programming, Management and Integration*. Greenwich: Manning, první vydání, 2002, 352 s., iSBN 1-930110-40-5.

[10] Harrison, R.: Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4513

[11] Hashimi, S. Y.; Komatineni, S.; MacLean, D.: *Pro Android 4*. New York: Apress, první vydání, c2012, 1020 s., iSBN 978-1-4302-3930-7.

[12] Howes, T.; Smith, M.; Good, G.: *Understanding and deploying LDAP*. Boston: Addison-Wesley, první vydání, 2003, 899 s., iSBN 0-672-32316-8.

[13] Legg, S.: Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4517

[14] Mednieks, Z. R.: *Programming Android*. Sebastopol: O'Reilly, první vydání, 2011, 482 s., iSBN 978-1-449-38969-7.

[15] Pascual, V. S.; Xhafa, F.: Evaluation of contact synchronization algorithms for the Android platform. *Mathematical and Computer Modelling*, ročník 57, č. 11–12, 2013: s. 2895 – 2903, ISSN 0895-7177, information System Security and Performance Modeling and Simulation for Future Mobile Networks.
URL http://www.sciencedirect.com/science/article/pii/S0895717711008132

[16] Sciberras, A.: Lightweight Directory Access Protocol (LDAP): Schema for User Applications. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4519

[17] Sermersheim, J.: Lightweight Directory Access Protocol (LDAP): The Protocol. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4511

[18] Six, J.: *Aplication Security for the Android Platform*. Cambridge: O'Reilly, první vydání, 2012, 114 s., iSBN 1-4493-1507-0.

[19] Smith, M.: Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4515

[20] Smith, M.: Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4516

[21] Zeilenga, K. D.: Lightweight Directory Access Protocol (LDAP): Directory Information Models. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4512

[22] Zeilenga, K. D.: Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4518

[23] Zeilenga, K. D.: Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names. [online], 6-2006, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc4514

[24] Zeilenga, K. D.: Lightweight Directory Access Protocol version 2 (LDAPv2) to Historic Status. [online], 3-2003, [cit. 27. května 2014].
URL http://tools.ietf.org/html/rfc3494

Příloha A

Obsah CD

./doc/	- programová dokumentace
./src/	- zdrojové soubory android aplikace syncContact
./openLDAP/src	- zdrojové soubory OpenLDAP
./openLDAP/ldif	- schéma GoogleContact a~inizializační soubory
./tex/	- zdrojová data technické zprávy
./dip-xsendl00.pdf	- technická zpráva
./MANUAL	- manuál
./README	- základní informace

Příloha B

Návod OpenLDAP

Pro použití aplikace *SyncContact* je nutné mít správně nastavený server. Aplikace byla vyvíjena a testována na serveru OpenLDAP verze 2.4.31 a na operačním systému Ubuntu 13.10 64.bit. Tato část popisuje jak nainstalovat OpenLDAP a jak provést základní nastavení.

Instalace OpenLDAP

Instalace LDAP serveru je možné provést několika způsoby. Na CD v adresáři `./openLDAP/src/` jsou zdrojové kódy serveru ve verzi 2.4.31. Je tedy možné provést instalaci ze zdrojových kódu nebo následujícím způsobem:

1. `$ sudo apt-get install slapd ldap-utils`

Při instalaci budete vyzváni k zadání hesla, například `.synccontact`.

Nastavení `ldap.conf`. Údaje zadávejte dle svého zvoleného DN a adresy serveru, kde instance poběží.

2. `$ sudo vim /etc/ldap/ldap.conf`
   `BASE dc=synccontac,dc=xsendl00,dc=cz`
   `URI ldap://192.168.0.103`

Rekonfigurace OpenLDAP. Po níže uvedeném příkazu se zobrazí obrazovky. Zadejte údaje, níže vypsané jsou jen orientační.

```
2.  $ sudo dpkg-reconfigure slapd
    -> Configuring slapd            : 'NO'
    -> Enter DN name                : 'synccontact.xsendl00.cz'
    -> Organizationname             : 'synccontact'
    -> Admin pass                   : 'synccontact'
           reentered                : 'synccontact'
    -> database ..                  : 'OK'
            backend                 : 'HDB'
    -> do you want to removed...    : 'YES'
    -> move old database            : 'YES'
    -> Allow LDAPv2                 : 'NO'
```

Příkazem `$ ldapsearch -x` je možné nastavení serveru otestovat.

Přidání nového schéma

Na přiloženém CD jsou ve složce `./openLDAP/ldif/` umístěné soubory pro základní nastavení OpenLDAP. Konkrétně jde o schéma *GoogleContact*, které je nezbytné pro zálohování kontaktů.

Nové schéma *GoogleContact* lze například přidat tímto způsobem. Nejprve vytvoříme soubor `export.conf` jeho obsah je následující:

3. `$ vim export.conf`

```
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/misc.schema
include /etc/ldap/schema/googleContact.schema
```

Obsahem souboru jsou jednotlivá schémata s plnou cestou jejich umístění. V případě potřeby souborům upravte cestu nebo je odstraňte. Pouze `core.schmema` zde musí zůstat, obsahuje základní parametry a třídy. Nezapomeňte nakopírovat soubor `./openldap/ldif/googleContact.schema` do správného adresáře.

5. `$ sudo cp googleContact.schema /etc/ldap/schema/`

Vytvoříme složku `export.d`

6. `$ mkdir export.d`

Nyní je vše připraveno k vygenerování *LDIF* souboru ze schématu googleContact

7. `$ slaptest -f export.conf -F export.d`

Tímto krokem se v adresáři `export.d` vygenerovalo několik souborů. Nás zajímá pouze `export.d/cn=config/cn=schema/cn={5}googleContact.ldif`

Upravte tři řádky v hlavičce podle následující předlohy:

8. `$ vim export.d/cn\=config/cn\=schema/cn\=\{5\}googlecontact.ldif`

```
dn: cn=googleContact,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: googleContact
```

Pro lepší čitelnost je možné smazat několik posledních řádků na konci souboru:

9. `$ vim export.d/cn\=config/cn\=schema/cn\=\{5\}googlecontact.ldif`

```
structuralObjectClass:
entryUUID:
creatorsName:
createTimestamp:
entryCSN:
modifiersName:
modifyTimestamp:
```

Posledním krokem se schéma přidá do stromové struktury LDAP serveru.

```
10.  $ sudo ldapadd -Y EXTERNAL -H ldapi:///
        -f export.d/cn\=config/cn\=schema/cn\=\{5\}googlecontact.ldif
```

Při úspěchu se zobrazí: `adding new entry "cn=googleContact,cn=schema,cn=config"` Krokem 10. bylo přidáno nové schéma. Nyní již je možné vytvořit základní stromovou strukturu. Ta je v podstatě velice jednoduchá, jak bylo popsáno v samotné práci. Pro její vytvoření stačí zadat následující příkaz:

```
11.  $ ldapadd -x -D 'cn=admin,dc=synccontact,dc=xsendl00,dc=cz'
               -W -f init.ldif
```

Bude vyžadováno heslo pro LDAP server. Nyní je server připraven pro nezabezpečenou komunikaci. V případě využití SSL/TLS nebo StartTLS je nutné vytvořit certifikáty.

Příloha C

Třída GoogleContact

Schéma třídy *GoogleContact*, které je využita při ukládání kontaktů z Android zařízení v adresářové struktuře serveru. Níže uvedené schéma obsahuje pouze hlavičku a definici třídy, zbylá část (parametry) jsou k dispozici na přiloženém CD.

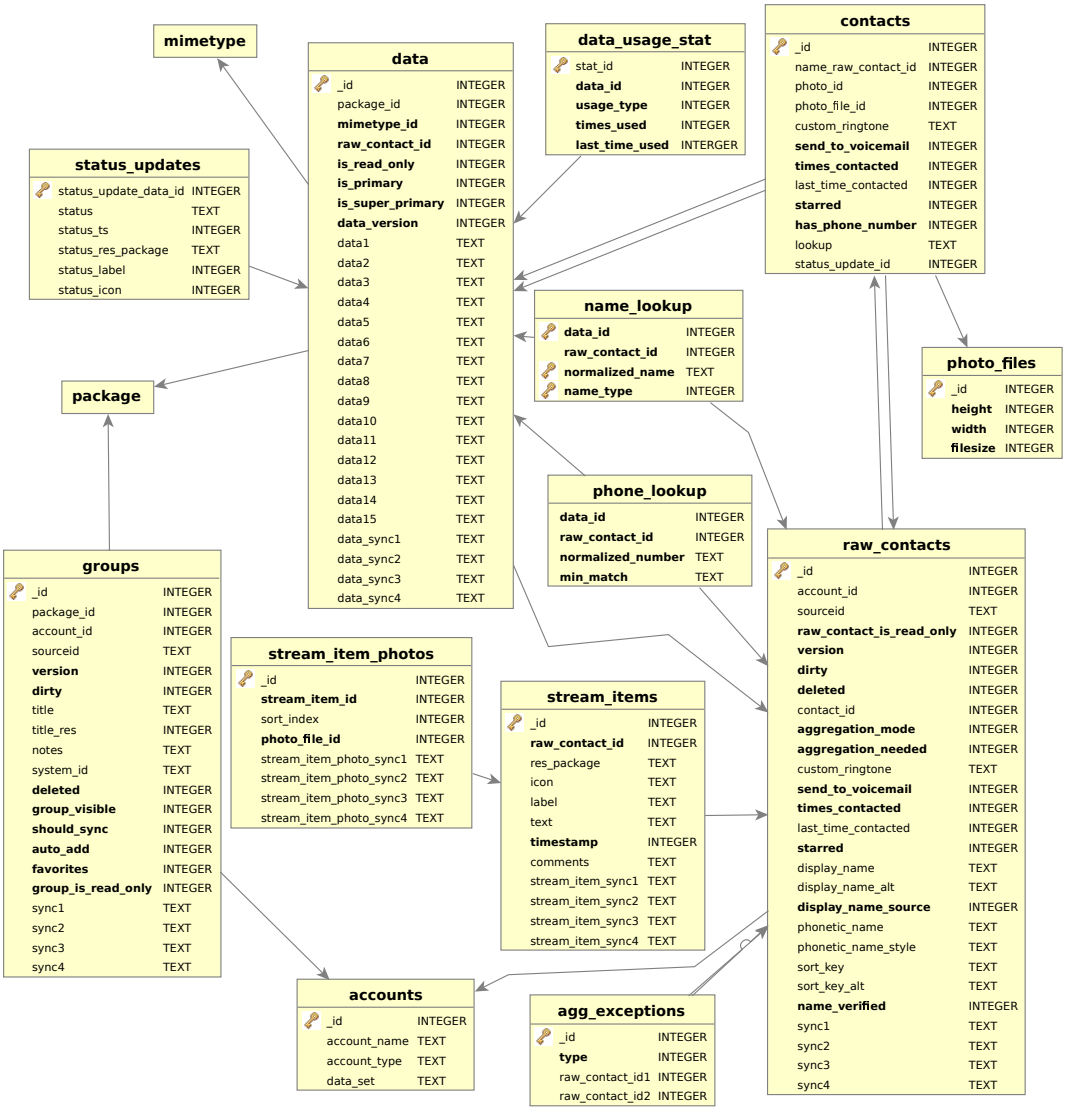
```
objectidentifier syncSchema 1.3.6.1.4.1.43624
objectidentifier syncAttrs syncSchema:3
objectidentifier syncOCs syncSchema:4

objectclass ( syncOCs:1
  NAME 'googleContact'
  SUP top
  STRUCTURAL
  MUST uuid
  MAY ( homeMail $ workMail $ otherMail $ mobileMail $ eventOther $ eventBirthday
        $ eventAnniversary $ identityText $ identityNamespace $ imHomeAim
        $ imHomeGoogleTalk $ imHomeIcq $ imHomeJabber $ imHomeMsn $ imHomeNetmeeting
        $ imHomeQq $ imHomeSkype $ imHomeYahoo $ imWorkAim $ imWorkGoogleTalk
        $ imWorkIcq $ imWorkJabber $ imWorkMsn $ imWorkNetmeeting $ imWorkQq
        $ imWorkSkype $ imWorkYahoo $ imOtherAim $ imOtherGoogleTalk $ imOtherIcq
        $ imOtherJabber $ imOtherMsn $ imOtherNetmeeting $ imOtherQq $ imOtherSkype
        $ imOtherYahoo $ nicknameDefault $ nicknameOther $ nicknameMaiden
        $ nicknameShort $ nicknameInitials $ notes $ organizationWorkCompany
        $ organizationWorkTitle $ organizationWorkDepartment
        $ organizationWorkJobDescription $ organizationWorkSymbol
        $ organizationWorkPhoneticName $ organizationWorkOfficeLocation
        $ organizationWorkPhoneticNameStyle $ organizationOtherCompany
        $ organizationOtherTitle $ organizationOtherDepartment
        $ organizationOtherJobDescription $ organizationOtherSymbol
        $ organizationOtherPhoneticName $ organizationOtherOfficeLocation
        $ organizationOtherPhoneticNameStyle $ phoneAssistant $ phoneCallback
        $ phoneCar $ phoneCompany $ phoneFaxHome $ phoneFaxWork $ phoneHome
        $ phoneISDN $ phoneMain $ phoneMMS $ phoneMobile $ phoneOther $ phoneOtherFax
        $ phonePager $ phoneRadio $ phoneTelex $ phoneTTYTDD $ phoneWork
        $ phoneWorkMobile $ phoneWorkPager $ relationAssistant $ relationBrother
        $ relationChild $ relationDomesticPartner $ relationFather $ relationFriend
        $ relationManager $ relationMother $ relationParent $ relationPartner
        $ relationRefferedBy $ relationRelative $ relationSister $ relationSpouse
        $ workSip $ homeSip $ otherSip $ phoneticMiddleName $ phoneticGivenName
        $ phoneticFamilyName $ familyName $ middleName $ givenName $ namePrefix
        $ nameSuffix $ homeStreet $ homePOBox $ homeCity $ homeRegion $ homePostalCode
```

```
        $ workStreet $ workPOBox $ workCity $ workRegion $ workPostalCode $ workCountry
        $ workFormattedAddress $ homeFormattedAddress $ workNeighborhood
        $ homeNeighborhood $ otherNeighborhood $ otherStreet $ otherCity $ otherPOBox
        $ otherRegion $ otherPostalCode $ otherCountry $ otherFormattedAddress
        $ websiteHomepage $ websiteBlog $ websiteProfile $ websiteHome $ websiteWork
        $ websiteFtp $ websiteOther $ displayName $ homeCountry $ accountTypePrevious
        $ accountNamePrevious $ synchonize $ deleted
        )
)
```

Příloha D

Databáze ContactProvider



Obrázek D.1: Vybrané tabulky z databáze *contacts2.db*

Příloha E

Návrh obrazovek

(a) Kontakty (b) Server

Obrázek E.1: Obrazovky 2.